Vision-based Perception For Autonomous Robotic Manipulation

Örebro Studies in Technology 93

Dinh-Cuong Hoang

Vision-based Perception For Autonomous Robotic Manipulation

Title: Vision-based Perception For Autonomous Robotic Manipulation

Publisher: Örebro University, 2021
www.oru.se/publikationer

Printer: Örebro University/Repro 2021/11

ISSN 1650-8580
ISBN 978-91-7529-414-8

Abstract

In order to safely and effectively operate in real-world unstructured environments where a priori knowledge of the surroundings is not available, robots must have adequate perceptual capabilities. This thesis is concerned with several important aspects of vision-based perception for autonomous robotic manipulation. With a focus on topics related to scene reconstruction, object pose estimation and grasp configuration generation, we aim at helping robots to better understand their surroundings, to avoid undesirable contacts with the environment and to accurately grasp selected objects.

With the wide availability of affordable RGB-D cameras, research on visual SLAM (Simultaneous Localization and Mapping) or scene reconstruction has made giant strides in development. As a key element of an RGB-D reconstruction system, a large number of registration algorithms have been proposed in the context of RGB-D Tracking and Mapping (TAM). The state-of-the-art methods rely on color and depth information to track camera poses. Besides depth and color images, semantic information is now often available due to the advancement of image segmentation driven by deep learning. We are interested to explore to what extent the use of semantic cues can increase the robustness of camera pose tracking. This leads to the first contribution of this dissertation. A method for reliable camera tracking using an objective function that combines geometric, appearance, and semantic cues with adaptive weights.

Beyond the purely geometric model of the environment produced by classical reconstruction systems, the inclusion of rich semantic information and 6D poses of object instances within a dense map is useful for robots to effectively operate and interact with objects. Therefore, the second contribution of this thesis is an approach for recognizing objects present in a scene and estimating their full pose by means of an accurate 3D semantic reconstruction. Our framework deploys simultaneously a 3D mapping algorithm to reconstruct a semantic model of the environment, and an incremental 6D object pose recovery algorithm that carries out predictions using the reconstructed model. We demonstrate that we can exploit multiple viewpoints around the same object to achieve robust and stable 6D pose estimation in the presence of heavy clutter and occlusion.

The methods taking RGB-D images as input have achieved state-of-the-art performance on the object pose estimation task. However, in a number of cases, color information may not be available — for example, when the input is point cloud data from laser range finders or industrial high-resolution 3D sensors. Therefore, besides methods using RGB-D images, studies on recovering the 6D pose of rigid objects from 3D point clouds containing only geometric information are necessary. The third contribution of this dissertation is a novel deep learning architecture to address the problem of estimating the 6D pose of multiple rigid objects in a cluttered scene, using only a 3D point cloud of the scene as an input. The proposed architecture pools geometric features together using a self-attention mechanism and adopts a deep Hough voting scheme for pose proposal generation. We show that by exploiting the correlation between poses of object instances and object parts we can improve the performance of object pose estimation.

By applying a 6D object pose estimation algorithm, robots can perform grasping known objects where the 3D model of objects is available and a grasp database is pre-defined. What if we want to grasp novel objects? The fourth contribution of this thesis is a method for robust manipulation of novel objects in cluttered environments. we develop an end-to-end deep learning approach for generating grasp configurations for a two-finger parallel jaw gripper, based on 3D point cloud observations of the scene. The proposed model generates candidates by casting votes to accumulate evidence for feasible grasp configurations. We exploit contextual information by encoding the dependency of objects in the scene into features to boost the performance of grasp generation.

Acknowledgements

First and foremost, I am extremely grateful to my supervisor – Todor Stoyanov for giving me the opportunity to do this research under his supervision, for providing me with every bit of guidance, assistance, and expertise. I could not have asked for a better supervisor for my PhD studies. Thank you, Todor!

Next, I would like to thank my secondary supervisor – Achim Lilienthal, whose expertise was invaluable in formulating the research questions. Many thanks for all suggestions, great ideas, and detailed revisions of my first three papers.

Also, I am indebted to Johannes Andreas Stork who has been supervised me for the latter half of my studies. Although he is not my official supervisor, his supervision is of great importance to my PhD studies. Thank you for the discussions and ideas we have shared, for reading and commenting on the manuscripts.

Thanks, also, to all the colleagues at the Autonomous Mobile Manipulation lab, at the Center for Applied Autonomous Sensor Systems (AASS), and friends at Orebro University.

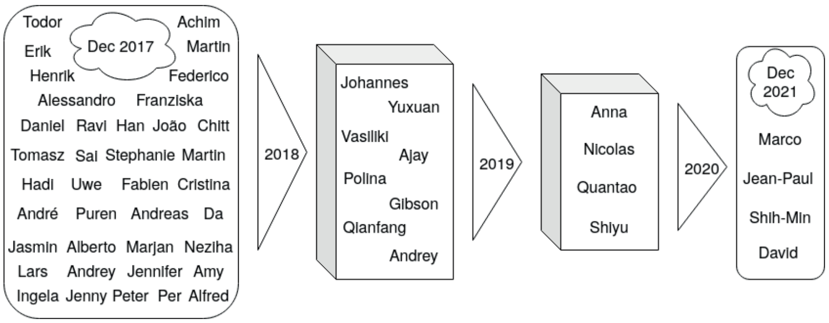


Figure 1: Thank you all!

Finally, I wish to thank my parents, younger brother, and relatives for their love and support.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

This year, the term “robot” celebrates its 100th anniversary, first appearing in Karel Capek’s play R.U.R. (Rossum’s Universal Robots). The word “robot” derives from the Czech for “labor”. Since then, super-human robots have always been a compelling topic in science fiction books and movies. In fictional worlds, robots are sentient, far more efficient than their human counterparts, and able to deal with all environment conditions. The robots of science fiction like Optimus Prime or R2-D2 formed the perception of robots in ordinary people’s minds, but also triggered their interest in science and inspired advancements of technology in the real world. Over the past decades, our concept of robots has evolved from fictional technologies to real-world machines integrated into manufacturing settings and many other areas. It is routine to see robot manipulators being deployed in auto-factories for repetitive tasks such as welding, painting, cutting, assembly, handling, etc. Indeed, robots have transformed the manufacturing industry, especially automotive factories. However, the fictional robots remain a long-term goal rather than a reality. Despite the development of AI and powerful sensors, robots today still perform far worse on many skills that come naturally to humans. Manipulating objects in an uncontrolled setting is a typical example. Even the most advanced robot would be unable to get you a cup of coffee smoothly and autonomously.

Humans have a fantastic ability to interact with and manipulate objects in the physical world. By just taking a glance at the scene, we can immediately, almost instinctively, localize the objects of interest and move our bodies in complex ways to grab and pick them up. This is contrary to the state-of-the-art autonomous robots of today, since manipulation is hard, especially in open environments. This fact may seem strange, as we often see robots move with blurring speed and perform specific tasks with extraordinary precision and repeatability in factories. However, they are only skilled at specific tasks in carefully controlled settings such as industrial assembly setups. In unstructured environments where robots do not have complete knowledge about their surrounding, they are much less capable. Inspired by humans’ remarkable ability to perceive the environment and manipulate objects, perception for manipulation has become one of the most studied topics. This thesis focuses on vision-based perception to help robots better understand their surroundings, avoid undesirable contact with the environment, and accurately grasp selected objects.

1.1 Challenges of Vision-based Robot Manipulation

In order to safely and effectively operate in real-world unstructured environments where a priori knowledge of the surroundings is not available, robots must have adequate perceptual capabilities. Vision technologies enable robots to sense the world and understand the state of the world. However, vision sensors can only provide partial knowledge of the surroundings. To expand the knowledge, robots can move cameras and obtain information (images) from different locations. Then the acquired information needs to be fused into a consistent scene representation. This problem is referred to as scene reconstruction and has been the target of several decades of research. The task of reconstructing the surrounding environment relates to several sub-problems including camera selection, calibration, camera pose estimation, data fusion, scene representation. As one of the most challenging pieces of the problem, camera pose tracking is difficult due to measurement noise, illumination variation, the limit of computing resources in practice, undesired elements such as planar surfaces, and more. The most widely investigated class of scene reconstruction systems relies on depth and color information to estimate camera pose. However, tracking failures can occur in case there are not enough features to fully constrain all 6DOF of the camera pose. In Chapter 3, we discuss state-of-the-art methods for camera tracking and propose a robust camera pose estimation approach.

The purely geometric map of the surrounding environment produced by scene reconstruction is not sufficient to enable robots to operate safely and effectively in applications with a high demand for flexibility. For instance, automated picking and manipulation of boxes and other types of goods requires information about the position and orientation of objects. Robots need to have the capability of understanding the 3D structure and semantics of their environment. Thus, fusing semantic along with geometric information and providing the position and orientation of target objects within a 3D reconstructed map is a promising approach to enable robots to better understand a 3D scene. It is especially important for mobile manipulation in which robots simultaneously navigate in unknown environments and pick objects. To accurately grasp selected objects and avoid collisions with neighboring obstacles in the workspace, the reconstruction process needs to produce a high-quality map of the working environment. The addition of semantic information and object poses enables a much greater range of functionality than geometry alone. However, recognizing objects and recovering their poses is a significant challenge, because the target objects can be any form, shape, and have six degrees of freedom (DOF). In addition, the targets may be arranged in a challenging manner in the scene. Because there are many objects with the same shape and appearance, it is not easy to recognize the right object and place targets at the defined position, especially in cluttered environments and under large occlusions. In Chapter 4, we present an approach for recognizing objects present in a scene and estimating their full pose by means of an accurate 3D semantic reconstruction.

Providing pose of objects in the 3D space enables robots to meaningfully interact with the objects as well as to deftly and safely move in open environments. Deep learning-based methods taking RGB or RGB-D images as input have achieved promising results on the object pose estimation task. Nonetheless, these methods require large amounts of labeled training data. Collecting images of objects from the real world under various conditions and annotating the images with 6D object poses is time-consuming and requires a significant human effort. A promising alternative is the use of synthetic data for training such deep neural networks. Yet, it is still difficult for methods requiring color information where the domain gap between synthetic training and real test images is severe. In addition, in a number of cases, color information may not be available — for example, when the input is point cloud data from laser range finders or industrial high-resolution 3D sensors. Therefore, besides methods using RGB or RGB-D images, studies on recovering the 6D pose of rigid objects from 3D point clouds containing only geometric information are necessary. In Chapter 5, we present a novel approach to address the problem of estimating the 6D pose of multiple rigid objects in a cluttered scene, using only a 3D point cloud of the scene as an input. Compared to color images, the domain gap between the synthetic and real data is considerably smaller for 3D point clouds. In addition, synthesizing data with only geometric information is less expensive in the terms of time and hardware storage as there is no texture or illumination present in the data. This low cost allows us to scale it up to a large number of objects, which is often desired in practical applications.

Previously, we discussed the challenges of the object pose estimation problem. Assuming that a robot had an accurate pose of objects, then how can a robot arm grasp an object? Given the pose of objects, a model of the object to be grasped can be aligned to measured data. A set of grasps is then selected from a database of pre-computed grasps. Given a gripper configuration, the robot can start to plan its motion and perform a grasp. However, predicting the pose of novel objects is not possible, as for 6D object pose estimation we assume that the 3D model of the object is available and crucially that the object coordinate system is defined in the 3D space of the model. This leads to a new problem of predicting grasps for novel objects. We discuss and propose an approach to address this problem in Chapter 6.

1.2 Outline

The following chapters will explore some aspects of the most crucial components of a vision-based perception system for robot manipulation. The rest of this thesis is organized as follows.

Chapter 2 makes a broad introduction to the field of vision-based perception for robot manipulation, and the sub-problems that it contains.

Chapter 3 discusses the problem of registration of RGB-D images. This chapter introduces a reliable camera tracking method and compares the proposed approach against current algorithms on a newly collected warehouse dataset.

Chapter 4 begins with a discussion about the problem of 6D object pose estimation using an RGB-D camera. It then introduces a method for multi-view 6D object pose using accurate semantic reconstruction and shows experimental results on the YCB-Video dataset [133] and the warehouse dataset.

Chapter 5 discusses the problem of recovering the 6D pose of rigid objects from 3D point clouds containing only geometric information. This study introduces a novel deep learning architecture to address the problem of estimating the 6D pose of multiple objects in a cluttered scene given a point cloud. The proposed approach is evaluated on data from the Siléane dataset [10] and the Fraunhofer IPA dataset [64]. A comparison against the most closely related works is also performed here.

Chapter 6 discusses the problem of grasp detection that generates grasp configurations directly from 3D point clouds for robotic manipulation without estimating object pose. This chapter proposes and evaluates an end-to-end context-aware grasp detection network for generating collision-free grasps. To build robustness to occlusion, the proposed network generates candidates by casting votes to accumulate evidence for possible grasps. The proposed approach is evaluated and compared with other state-of-the-art methods on the public dataset GraspNet-1Billion [29].

Chapter 7 concludes this dissertation and summarizes the major contributions and directions of future research.

1.3 Contributions

The major contributions of this thesis, as outlined in the previous section, can be summarized as follows:

A method for reliable camera tracking and state-of-the-art surface reconstruction using an objective function that combines geometric, appearance, and semantic cues with adaptive weights. (Chapter 3)

An incremental 6D object pose recovery algorithm that carries out predictions from multiple viewpoints around the same object to achieve robust object pose in the presence of heavy clutter and occlusion. (Chapter 4)

A novel end-to-end trainable artificial neural network architecture for estimating the 6D pose of multiple rigid objects in a cluttered scene. The approach uses only a 3D point cloud of the scene as an input and learns to abstract lower-level point features into high-level features by encoding the dependency between object parts and object instances. (Chapter 5)

An end-to-end deep learning approach for generating 6-DOF collision-free grasps of unknown objects, based on 3D point cloud observations of the scene. To build robustness to occlusion, the proposed model generates candidates by casting votes to accumulate evidence for feasible grasp poses. The approach exploits contextual information by encoding the dependency between objects in the scene into features to boost the performance of grasp generation. The contextual information enables the model to increase the likelihood that the generated grasps are collision-free. (Chapter 6)

1.4 Publications

Parts of this work have appeared previously in a number of journal and conference papers. The following list summarizes all the publications accomplished during the course of working towards this thesis, as well as the precise chapters of this work that each article contributed to.

- Hoang, D. C., Stoyanov, T., & Lilienthal, A. J. (2019, September). Object-RPE: Dense 3D Reconstruction and Pose Estimation with Convolutional Neural Networks for Warehouse Robots. *In 2019 European Conference on Mobile Robots (ECMR)* (pp. 1-6). IEEE.
  Main part of Chapters 3 and 4

- Hoang, D. C., Lilienthal, A. J., & Stoyanov, T. (2020). Object-RPE: Dense 3D reconstruction and pose estimation with convolutional neural networks. *Robotics and Autonomous Systems (RAS)*, 133, 103632.
  Main part of Chapters 3 and 4

- Hoang, D. C., Lilienthal, A. J., & Stoyanov, T. (2020). Panoptic 3D Mapping and Object Pose Estimation Using Adaptively Weighted Semantic Information. *IEEE Robotics and Automation Letters (RAL)*, *5*(2), 1962-19. (accepted and presented at IEEE International Conference on Robotics and Automation (ICRA 2020)).
  Main part of Chapters 3 and 4
- Hoang, D. C., Stork, J. A, & Stoyanov, T. Voting and Attention-based Pose Relation Learning for Object Pose Estimation from 3D Point Clouds. *IEEE Robotics and Automation Letters (RAL)*. Under review.
  Main part of Chapter 5
- Hoang, D. C., Stork, J. A., & Stoyanov, T. Context-Aware Grasp Detection for Target Objects in Cluttered Scenes Using Deep Hough Voting. *IEEE International Conference on Robotics and Automation (ICRA 2022)*. Under review.
  Main part of Chapter 6

Chapter 2
Background

2.1 Dense RGB-D Reconstruction

Three-dimensional (3D) scene reconstruction from images is an important research topic that derives from computer graphics and computer vision. It has drawn considerable attention from the industrial and research communities and has become a core technology of a wide variety of fields. In robotics, simultaneous localization and mapping (SLAM) using RGB-D cameras (also can be referred to as online dense RGB-D reconstruction) – is one of the most important factors for scene understanding, planning and interaction. To achieve effective and autonomous operation in unstructured environments, robots need to perceive the 3D world and infer information about the environment. The reconstructed scene models together with 3D data processing algorithms would enable robots to identify features in their surroundings, recognize relevant objects and acquire task-relevant knowledge. With such importance placed on the scene modeling, visual SLAM has been extensively studied by the robotics community in recent years. We have seen a solid progress towards achieving real-time, dense reconstruction of complex scenes using RGB-D cameras. Fig. 2.1 shows a reconstructed model of an office room containing over 2.5 million colored vertices. In this section, we provide the background as well as an overview of recent advances in dense RGB-D reconstruction techniques.

2.1.1 RGB-D Data

An RGB-D image is a combination of a colored image (RGB) and its corresponding depth image (D). In depth images, each pixel relates to a distance between the image plane and the corresponding object. RGB and Depth images are often aligned, in the sense that pixels in the appearance and geometric channels correspond to the projection of the same physical point on to the camera plane. Fig. 2.2 shows an example of an RGB image and its corresponding depth image. A variety of techniques have been developed to augment color images with geometric information. These include structured light, time of flight (ToF),

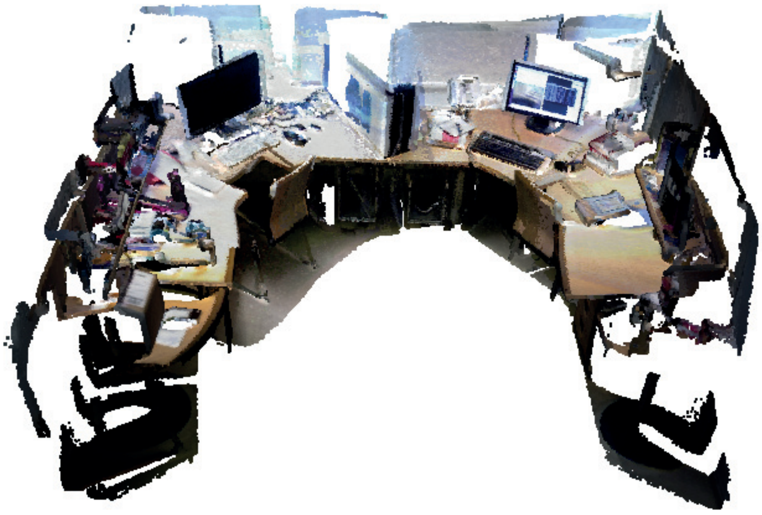


Figure 2.1: An example of a 3D model of an office room at Orebro University. The model reconstructed by ElasticFusion framework [129] contains over 2.5 million colored vertices.

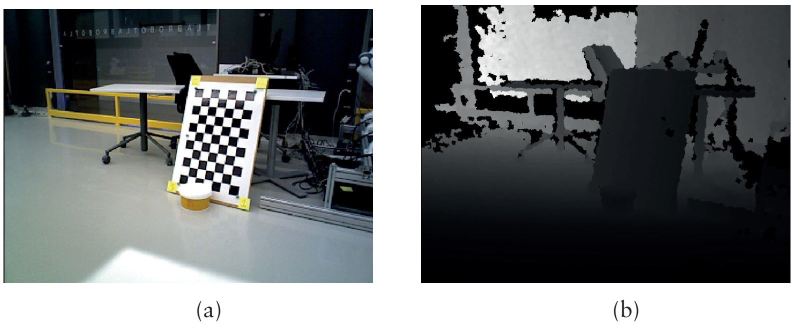


Figure 2.2: Example of an RGB-D image captured by a Xtion Pro Live camera. (a) Color image; (b) Depth image.

stereo vision, active infrared stereo cameras (AIRS). They also can be divided in two main approaches: active and passive depth sensing. Structured light methods make use of a projection device to project known patterns into the scene and calculate the depth based on deformation of pattern when striking objects surfaces. Sensors using this technique are called active systems because they actively modify the scene to obtain surface information of the objects. ToF cameras are active systems as well, which acquire depth information by measuring the round trip time of an emitted light pulse [30]. Different from the above two types of depth sensors, stereo cameras simulate human 3D vision to perceive depth based on the input of multiple conventional monochrome or color cameras. They are considered as passive systems due to to the fact that these sensors do not modify the scene to obtain the scene depth. The active infrared stereo cameras (AIRS) are an extension of the passive stereo sensors using an infrared stereo camera pair with pattern projective texturing the scene.

The above depth sensing techniques already date back several decades. However, RGB-D cameras only became widely available and affordable to a large user base since 2010 when Microsoft launched its first commodity depth sensor, known as Kinect v1. The first generation of Kinect incorporates a structured light based depth sensor and generates a 640x480 depth map at 30Hz. Kinect gained wide acceptance for applications in robotics and other fields because of its reasonable accuracy at a very affordable price, adequate resolution as well as real-time rates. Subsequently, other depth cameras using different depth sensing techniques were introduced, such as Kinect v2 (Microsoft), Xtion Pro Live (Asus), Astra (Orbbec), RealSense (Intel), etc. A comparison of common sensors is shown in Table 2.1. Similar to Kinect v1, Xtion PRO Live and Astra cameras are designed for short-range measurements using structure light technology. Generally, only a depth within 3.5 m is able to be used for reconstruction. The measurement accuracy of these structure light-based RGB-D devices decreases with the increase of the measurement distance. In contrast, ToF-based Kinect v2 the depth accuracy is almost constant over different distances and it slightly extended the range of depth values [125, 67]. However, the depth map resolution of ToF cameras like Kinect v2 does not reach a video graphic array resolution of 640x480 pixels. Kinect v2 has a resolution of 512x424 instead of 640x480, but the images (RGB, depth, and infrared) obtained with Kinect v2 were of better quality than those with Kinect v1. For higher resolution, Intel released RealSense D435 cameras with AIRS technology provides a resolution of 1920x1080 in RGB image and 1280x720 in the depth image. RealSense sensors are able to provide a robust solution in both indoor and outdoor scenarios. The advent of affordable consumer grade RGB-D cameras led to significant advances in dense 3D reconstruction. To evaluate scene modeling systems on RGB-D data, a number of RGB-D benchmarks have been introduced.

TUM Dataset: this benchmark [112] is one of the most popular datasets for the evaluation of RGB-D SLAM systems. The dataset covers a large variety of scenes and camera motions and provides sequences for debugging with slow

Table 2.1: Comparison of RGB-D sensor principles. The table is from the survey [32].

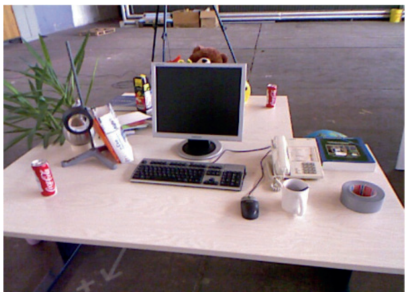
	Sensors	Advantages	Disadvantages
Structured Light	Kinect v1; Astra S; Xtion PRO Live	High precision at close range measurement	Low precision at long range measurement; heavily affected by illumination and light reflection
Time of Flight	Kinect v2	High precision at long range measurement; less affected by illumination	Low image resolution; high power consumption; high hardware cost
Active Infrared Stereo Cameras	RealSense R200 D435	Small size; light weight; high image resolution	High computation cost; poor real-time performance

motions as well as longer trajectories with and without loop closures. It consists of 39 sequences recorded in office and industrial environments using the Kinect v1. Each sequence contains the color and depth images, as well as the ground-truth trajectory from the motion capture system. Fig. 2.3 shows a recorded scene and camera trajectory.

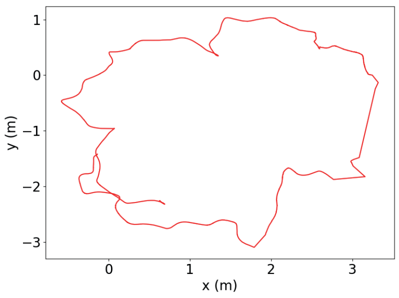
ScanNet [22] this dataset contains 2.5M RGB-D images in 1513 sequences acquired in 707 distinct spaces. The data is annotated with 3D camera poses, surface reconstructions, textured meshes, object-level semantic category labels, and aligned CAD models. To collect this dataset, the authors developed a scalable collection and annotation system with a low-cost RGB-D sensor similar to the Microsoft Kinect v1.

2.1.2 Surface Representation and Prediction

An important element of dense 3D reconstruction systems is the design of surface representation. A surface representation needs to be very efficient in fusing overlapping 3D points into a consistent global model to allow for real-time processing. Moreover, in order to increase the accuracy of scene reconstruction, the camera pose estimation process often requires dense surface prediction from the most up-to-date scene representation. This can be achieved by rendering a view of the currently reconstructed surface using a virtual camera. Therefore, the scene representation should allow an efficient rendering process. In addi-



(a)



(b)

Figure 2.3: Example of a sequence contained in TUM RGB-D dataset [112]: (a) recorded scene; (b) camera trajectory.

tion, it should also be adaptive for modeling both large or small environments without too much memory overhead. There are two widely used classes of representations for incremental dense reconstruction: voxel-based and surfel-based methods.

Voxel-Based Representation.

The truncated signed distance function (TSDF) is a voxel-based representation of 3D space which has been used in a large number of real-time dense 3D reconstruction methods [87, 128, 13]. The TSDF was originally introduced by Curless and Levoy [21] for the integration of range images. The TSDF has a set of desirable properties such as incremental updating, time and space efficiency, the ability to fill gaps. It is also well-suited for data-parallel algorithms. This is attractive due to the ability to enhance computational performance using GPGPU (General-purpose computing on graphics processing units).

In a signed distance function, the value at a given point is equal to the distance from the point to the nearest surface. By convention, for a true SDF the value is positive for points located in free space, and negative for points that are inside objects. The absolute value of the distance increases when advancing the surface. With the sign of distance value we can distinguish between points inside and outside of objects and extract the zero-crossing isosurface which represents the actual surface of objects. In practice, the distance values are truncated at pre-defined limits to lie between D_{min} and D_{max}. However, computing the true SDF is sophisticated and not well-posed from just measurements. Instead, a projective truncated signed distance function (TSDF) is usually used by estimating distances along the lines of sight of a range sensor.

Surfel-Based Representation.

Alternative to voxel-based representation, a reconstructed scene also can be represented with a set of surfels. This is the choice of scene representation in many reconstruction systems [3, 126, 62], especially it is one of the key components in the state-of-the-art RGB-D SLAM framework ElasticFusion [129]. Surfels are point primitives without explicit connectivity as defined in [95]. Each surfel might consist of different attributes such as position, color, normal, and others.

2.1.3 Reconstruction Frameworks

One of the earliest methods for real-time dense 3D reconstruction using RGB-D sensors (KinectFusion) was presented by Newcombe et al. [87]. KinectFusion fuses all of the depth data from a Kinect camera into a global voxel-based surface model (TSDF) of the observed scene. The TSDF is stored as a 3D voxel grid in GPU memory. We denote the global TSDF that contains a fusion of the registered depth measurements from frames 1...k as $\mathbf{S}_k(p)$, where $p \in \mathbb{R}^3$ is a given cell location (voxel). Two components are stored at each voxel including the current truncated signed distance value $S_k(p)$ and a weight $W_k(p)$:

$$\mathbf{S}_k(p) \mapsto [S_k(p), W_k(p)] \tag{2.1}$$

The weight W corresponds to the certainty of the distance value. The update of the weight and distance values is as follows:

$$S_k(p) = \frac{W_{k-1}(p)S_{k-1}(p) + W_{D_k}(p)S_{D_{k-1}}(p)}{W_{k-1}(p) + W_{D_k}(p)} \tag{2.2}$$

$$W_k(p) = W_{k-1}(p) + W_{D_k}(p) \tag{2.3}$$

With the currently reconstructed scene model available with continuous surface fusion, a surface prediction can be achieved by rendering the surface encoded in the zero level set $S_k = 0$ into a virtual camera. Fortunately, this can be done simply and efficiently on modern parallel hardware using implicit surface raycasting [92]. The output of the surface prediction is a predicted depth image that can be used in estimating camera pose. The sensor poses are obtained by registering the live depth frame to a predicted depth map from the global model using a variant of the iterative closest point (ICP) algorithm. Central to the functionality of KinectFusion is the use of highly parallel general purpose GPU techniques giving it the ability to perform tracking and mapping at the frame rate of the Kinect sensor.

However, due to the limitation of TSDF voxel model, KinectFusion is not able to create highly detailed maps of extended scale environments. Whelan et al. [128] introduced an extension to the KinectFusion algorithm that permits dense reconstruction of an unbounded extended area, called Kintinuous. Specifically, instead of restricting the tracking and surface reconstruction to the region around the point of initialization of the TSDF, Kintinuos permits the area mapped by the TSDF to move over time. They utilize a cyclical buffer data structure in implementation to continuously augment the reconstructed surface in an incremental fashion as the camera moves. Soon after Canelhas et al. [13] proposed a parallel algorithm for online RGB-D reconstruction that is practical for use on modern CPU hardware without a GPU. Unlike the works in [87, 128], their algorithm does not use predicted depth images from the current model for registration, but rather registers data directly to the TSDF model.

All the above frameworks rely solely on depth information for the estimation of camera pose. Using only depth data, tracking failure can occur in situations where the amount of characteristic features in the depth map is low. Steinbrucker et al. [111] introduced an energy minimization approach for RGB-D image registration that relies on color information instead. In comparison with geometric ICP, the authors reported that their method is more accurate in the regime of small camera motions. However, a reliance on color information alone for camera tracking suffers from a lack of visual features and does not exploit the rich geometric information contained within the scene. Whelan et al. [130] combined the color and depth information in the cost function so that all given information is used. They demonstrated that this combination increases the robustness of camera tracking across a variety of environments. This idea was further used in ElasticFusion [129] which fuses measurements and uses a surfel structure instead of a volumetric one for reconstruction.

In ElasticFusion, the scene model is represented by a cloud of surfels $\mathcal{M}^s$, where each surfel consists of a position $p \in \mathbb{R}^3$, normal $n \in \mathbb{R}^3$, colour $c \in \mathbb{N}^3$, weight $w \in \mathbb{R}$, radius $r \in \mathbb{R}$, initialisation timestamp t_0 and last updated timestamp t. The radius for each surfel is estimated to maximize distance between adjacent surfels and minimize visible holes. The radius r and weight w are initialized as:

$$r = \frac{1}{\sqrt{2}} \frac{d/f}{n_z} \tag{2.4}$$

$$w = \exp^{-\gamma^2/2\sigma^2} \tag{2.5}$$

where d is the depth, f is the focal length of the depth camera and n_z the z component of the estimated normal. γ is the normalized radial distance of the current depth measurement from the camera center. In accordance with the work [62] $\sigma = 0.6$ is derived empirically. The surfel-based scene representation is flexible, since surfel coordinates can be updated very efficiently for the whole reconstruction. We denote the prime superscript (*e.g.* p') for the newly associated measurement and the hat operator (*e.g.* $\hat{p}$) denotes the new updated value for a given surfel at the next time step. Then the update rules for each surfel component are detailed as follows:

$$\hat{p} = \frac{wp + w'p'}{w + w'} \tag{2.6}$$

$$\hat{n} = \frac{wn + w'n'}{w + w'} \tag{2.7}$$

$$\hat{r} = \frac{wr + w'r'}{w + w'} \tag{2.8}$$

$$\hat{w} = w + w' \tag{2.9}$$

The color attribute c is also updated in the same way. The surfel-based representation is highly adaptive as measurements at a higher resolution lead to denser point representation. Compared to voxel-based representation, surfel clouds are easier to handle thin objects. While raycasting is employed for rendering the voxel-based reconstruction, surfel-based methods render the explicit representation using a simple surface-splatting technique as introduced in [62]: they render overlapping, disk-shaped surface splats that are spanned by the model point's position p, radius r and normal n. Instead of using more refined surface-splatting techniques, EWA Surface Splatting [144] for example, they simply render opaque splats. The output of the splatted rendering is a predicted depth or color image which is used in Chapters 3 and 4. An in-depth system architecture diagram of ElasticFusion is shown in Fig. 2.4. The global model is simply an unordered list of surfels with associated attributes as presented above. Points evolve from unstable to stable status based on how often they are observed by the camera as illustrated in Fig. 2.5. Only surfels that are marked as active model surfels are used for camera pose estimation and depth map fusion.

2.2 Convolutional Neural Networks

Another foundational topic used throughout this thesis is deep learning. Chapter 3 introduces a segmentation framework with convolutional neural networks. Chapter 4 integrates Mask R-CNN [44] and DenseFusion [121] into our pipeline which use ResNet [45] as a feature extractor network. In the following, we provide the basic concepts of convolutional neural networks as well as some of the backbone network architectures we rely on.

Convolutional neural networks, also known as ConvNets or CNNs, are a specialized class of artificial neural network designed for working with data that has a grid-like structure [36]. A two-dimensional (2D) image can be viewed as a 2D grid of pixels which has spatial correlation between the neighborhood data points. In view of this, ConvNets is suitable for processing digital images, although they can also be used with one-dimensional and three-dimensional data. Indeed CNNs have been tremendously successful in computer vision tasks such as image classification, 2D object detection, semantic segmentation, etc.

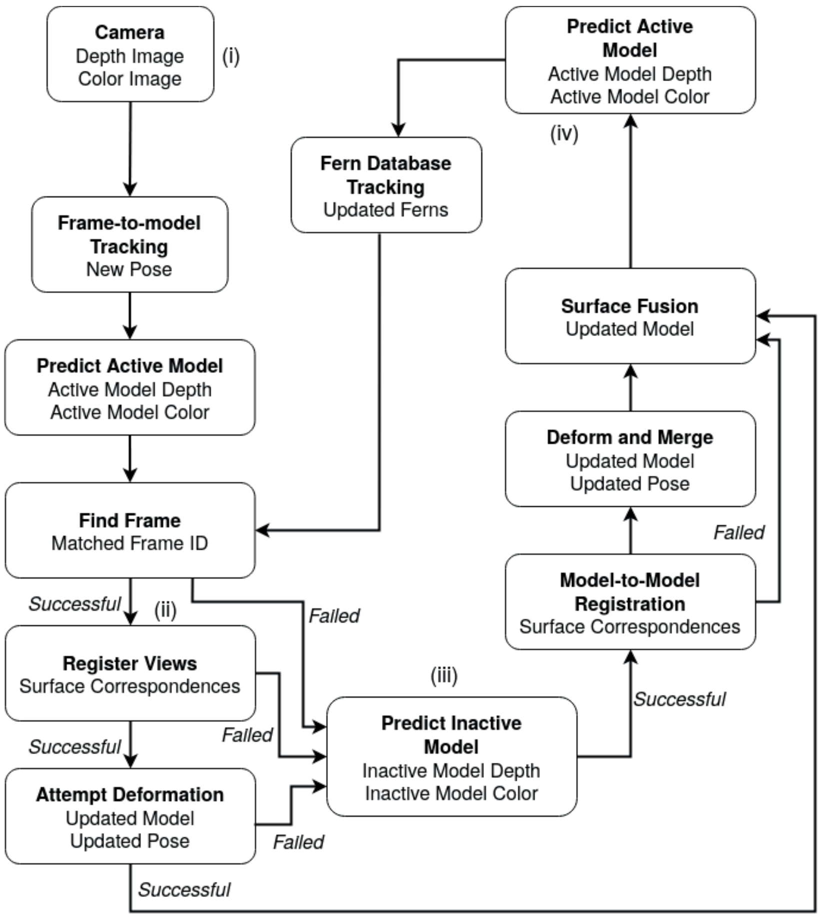


Figure 2.4: ElasticFusion architecture diagram. (i) The current depth and color images are aligned with the predicted models; (ii) Register views and attempt deformation to ensure local and global surface consistency; (iii) Labeling surfels that have not been seen in a period of time as inactive; (iv) Live camera data is fused with the latest updated model and an up to date prediction of the active model.

Central to the convolutional neural network is the mathematical operation convolution that gives the network its name. In mathematics, convolution is an operation on two functions $g(x)$ and $h(y)$ producing a third function and defined as

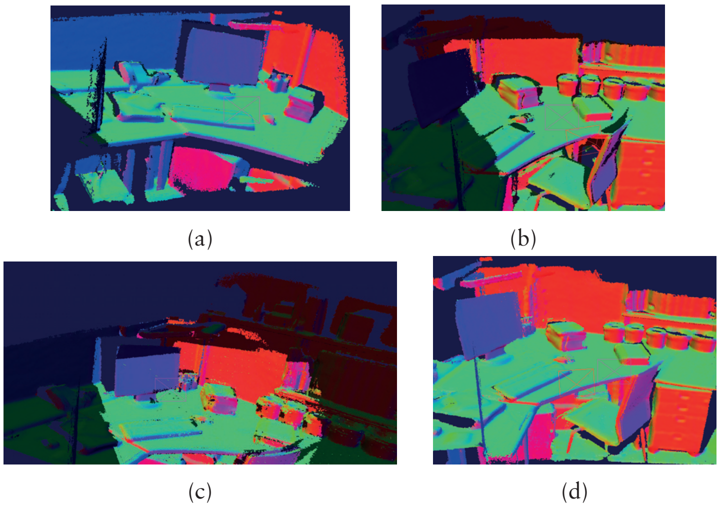


Figure 2.5: Example a sequence with active model colored by surface normal overlaid on the inactive model. (a) Initially all data is in the active model; (b) The area of map not seen recently is set to inactive; (c) The camera revisits the inactive area of the map, the inactive region then becomes active; (d) Final full map.

$$f(x) = \int g(y)h(x-y)dy \tag{2.10}$$

The convolution operation is typically denoted with an asterisk:

$$f(x) = (g * h)(x) \tag{2.11}$$

In ConvNets, the functions g, h, and f are often referred to as input, kernel, and feature map respectively. For image data, we use a 2D image as the input and a 2D kernel K:

$$F(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{2.12}$$

Because of commutative property of convolution, we can re-write Eq. 2.12 as

$$F(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m,j-n)K(m,n) \tag{2.13}$$

Fig. 2.6 illustrates convolution operation applied to a 2-D tensor. A typical layer of a convolutional network consists of three components as shown in Fig. 2.7. Since convolution is a linear operation and we want to realize rich and complex functions, the non-linearity stage often comes after the convolution stage to introduce non-linearity to the activation map. The final stage is pooling such as max pooling or average pooling to reduce the dimensions of the feature maps so that we can avoid overfitting as well as reduce the number of parameters and computation in the network.

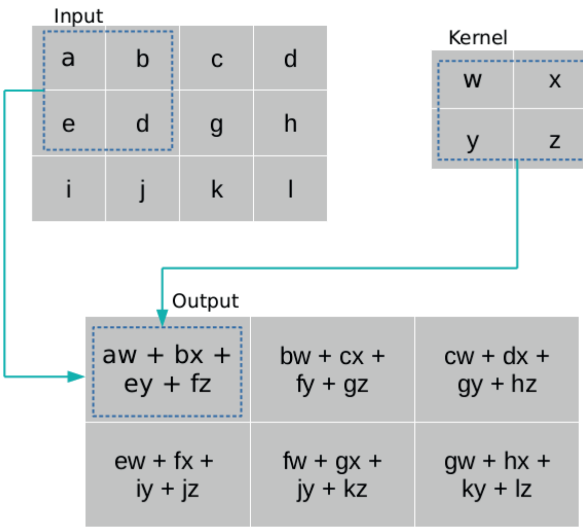


Figure 2.6: An example of 2-D convolution. The boxes with arrows indicate how the output is formed by applying the kernel to the corresponding region of the input tensor. Note that here we restrict the output to only positions where the kernel lies entirely within the image.

LeNet-5 [68] was one of the earliest CNN architectures that has the modern structure of ConvNets – stacked convolutional layers (as presented above) are followed by fully-connected layers. However, it was often surpassed by traditional machine learning methods such as support vector machine (SVM) or boosting algorithms due to lack of training data and computing power. Hence, it did not obtain enough attention at that time. The first viable CNN architecture that could outperform hand-designed features and SVMs was presented in 2012. AlexNet[65] achieved record-breaking results in ImageNet classification and outperformed all the entries by a large margin that year. The architecture of AlexNet comprises five convolutional and three fully-connected layers compared to three convolutional and three fully-connected in LeNet-5. By incorpo-

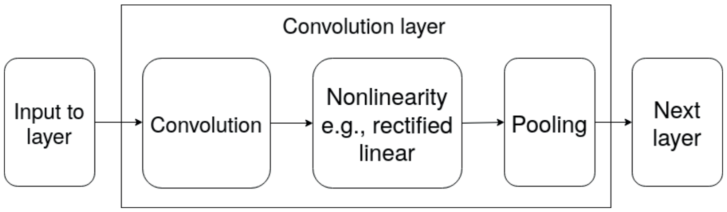


Figure 2.7: The components of a typical convolutional neural network layer [36]. In the first stage, convolutions are performed to produce a set of linear activations. Then, each linear activation is run through a nonlinear activation function. In the next stage, a pooling function is used to modify the output.

rating the nonlinearity function of ReLu (Rectified Linear Unit) instead of Tanh or Sigmoid, AlexNet was able to accelerate the training speed. The output of the last fully-connected layer is fed to a softmax function which produces a distribution over the 1000 class labels. There are 60 million trainable parameters in this network. To combat overfitting, they use data augmentation methods and dropout technique. Inspired by the success of AlexNet, researchers have put more effort to bring advancements in CNNs. Notably, Simonyan et al. proposed a deeper CNN model by adding more layers, named as VGG [109]. It was the first study proved that increasing depth of the network can improve the performance. VGG is made 19 layers deep and uses very small 3×3 kernels. The major drawback of VGG is the use of 138 million trainable parameters, which makes it computationally expensive.

To increase the depth of the network while keeping the computations to a constant level, Sermanet et al. [114] proposed GoogleNet model, also called Inception-v1 as there are v2, v3 and v4 later on. Inspired by human visual system, it introduced the new concept of inception architecture in CNN that allows visual information processed at various scales and then aggregated locally. To achieve this without a memory explosion, 1×1 convolutions are employed to compute reductions before the expensive 3×3 and 5×5 convolutions. In inception modules, 1x1 convolution is used as a dimension reduction module to reduce the computation that enables it to create deeper architecture. GoogLeNet has 22 layers in total including 9 inception modules and it replaces all fully connected layers with average pooling. This saves a lot of parameters. By going deeper with convolutions, this architecture was the winner at the ILSVRC 2014 image classification challenge [104]. VGG and GooLeNet confirm that "the deeper the better". The question is why don't they go deeper rather than 19 layers (VGG) or 22 layers (GoogLeNet). Adding more layers did not work for the researchers because of the well-known problem of vanishing gradients [6, 34]. While training deeper networks, during training accuracy gets saturated and then degrades rapidly. He et al. addressed the problem by introducing a deep residual learning framework, called ResNet [45]. The main contribution of ResNet is the residual block as illustrated in Fig. 2.8. The "skip connection" technique is the core of residual blocks. Rather than expect stacked layers to fit a desired underlying mapping H, we explicitly let these layers fit a residual mapping $F(x) := H(x) - x$. Then the original becomes $H(x) := F(x) + x$. The skip connections solve the problem of vanishing gradient by allowing information to skip layers which hurt the performance. ResNet architecture with 152 layers deep CNN won the 2015-ILSVRC competition [104]. A comparison of CNN architectures can be found in Table 2.2.

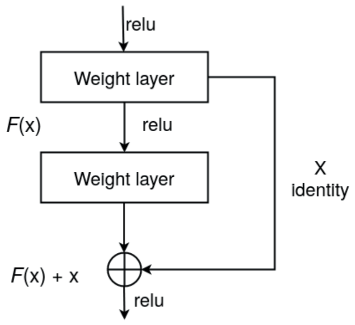


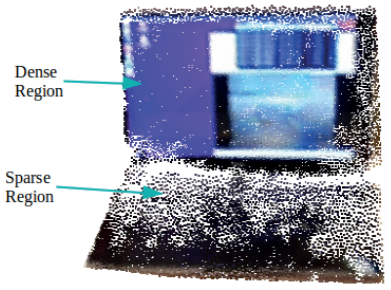
Figure 2.8: A residual block of deep residual network [45]. The "skip connection" technique is the core of residual blocks. This technique solves the problem of vanishing gradient by allowing information to skip layers which hurt the performance.

Table 2.2: The comparison of different CNN architectures on model size, classification error rate, and model depth.

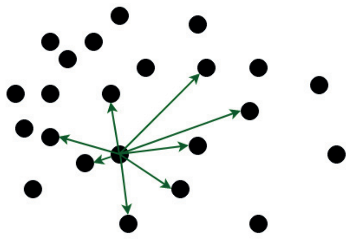
	Parameters	Error Rate	Depth
LeNet-5 [68]	0.060 million	MNIST: 0.95	5
AlexNet [65]	60 million	ImageNet: 16.04	8
VGG19 [109]	138 million	ImageNet: 7.3	19
GoogLeNet [114]	4 million	ImageNet: 6.7	22
ResNet-152 [45]	25.6 million	ImageNet: 3.6	152

2.3 Deep Learning on 3D Point Clouds

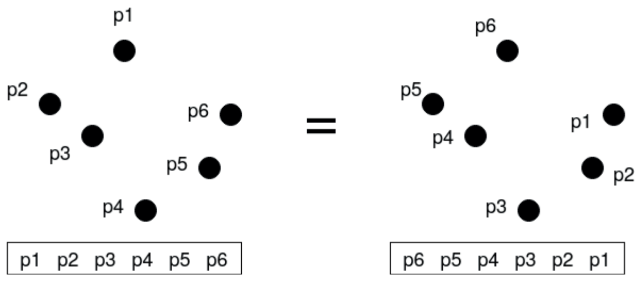
While Convolutional Neural Networks (CNN) are most commonly applied to 2D images, it is difficult for CNNs to handle point cloud data that is unordered, irregular and not on a structured grid as illustrated in Fig. 2.9. To reason about 3D geometric data, a variety of deep learning architectures for learning features from point cloud have been proposed [98, 99, 143, 124, 41]. In the next chapters, we will use PointNet [98] and PointNet++ [99] as backbone networks for feature extraction from 3D point sets. Therefore, in this section we detail the architecture of these two networks.



(a) Irregular.



(b) Unstructured.



(c) Unordered.

Figure 2.9: Properties of 3D point clouds: (a) Irregular – Sparse and dense regions; (b) Unstructured – Each point is independent and distance between neigh-boring points is not fixed;(c) Unordered – Point cloud are invariant to permutation.

PointNet [98] is a pioneering work in studying deep learning on 3D point clouds. Given an unordered point set $x_1, x_2, ...x_n$ with $x_i \in \mathbb{R}^3$, PointNet approximates a set funtion f to map the set of points to a vector:

$$f(x_1, x_2, ..., x_n) = \gamma \left(\max_{i=1,...,n} (h(x_i)) \right) \tag{2.14}$$

where max is a vector operator that takes n vectors as input and returns a new vector of the element-wise maximum, γ and h are multi-layer perceptron (MLP) networks. The network architecture of PointNet is visualized in Fig. 2.10. The input transform module consists of a shared MLP with layer output sizes (64, 128, 1024) on each point, a max pooling across points and two fully connected layers with output sizes (512, 256). All layers, except the last one, include ReLU and batch normalization. The output is a 3×3 matrix. Similarly, the feature transform module has the same architecture except that the output is a 64×64 matrix. The shared MLPs can be implemented as 1×1 convolutions to share weights and biases through all points. Note that 1×1 convolutions really are $1 \times 1 \times D$, where D is the feature dimensions. The key to PointNet is the use of max pooling layer as a symmetric function to select informative points and encode the reason for the selection. Thereby it is invariant to ordering.

Despite being able to encode the whole input point cloud into a global feature map. PointNet lacks the ability to capture local features. As an extension of PointNet, PointNet++ [99] is introduced to to resolve the limitation. It is a hierarchical neural network due to the fact that it processes the input point cloud in a metric space in a hierarchical fashion ((see Fig. 2.11)). In the first step points are grouped into local regions. Then we extract features of these regions using PointNet. The extracted features are further grouped into larger units and abstracted to higher level representations. This process is repeated until we obtain the features of the whole input point cloud. A high-level overview of the architecture is provided in Fig. 2.11. As we can see the core of PointNet++ network is set abstraction modules. Each module is composed by three key layers: sampling layer, grouping layer and PointNet layer. Given input point cloud $x_1, x_2, ..., x_n$, the sampling layer selects a subset of points $x_{i_1}, x_{i_2}, ..., x_{i_n}$ using iterative farthest point sampling (FPS). In the grouping layer, a ball query is used to find all points within a radius and form local regions. The input to this layer is a sampled point cloud with the size $N \times (d+C)$ and the coordinates of a set of centroids of size $N' \times d$. The output is N' clusters. Each cluster is a point set of size $K \times (d + C)$ corresponding to a local region and K is the number of points in the neighborhood of centroid points. The N' local regions of points are then fed into PointNet layer local pattern learning. By stacking several set abstraction levels, PointNet++ effectively learns deep point set features. Besides solving classification and semantic segmentation problems, PointNet++ is used as a backbone network for feature extraction in many other works.

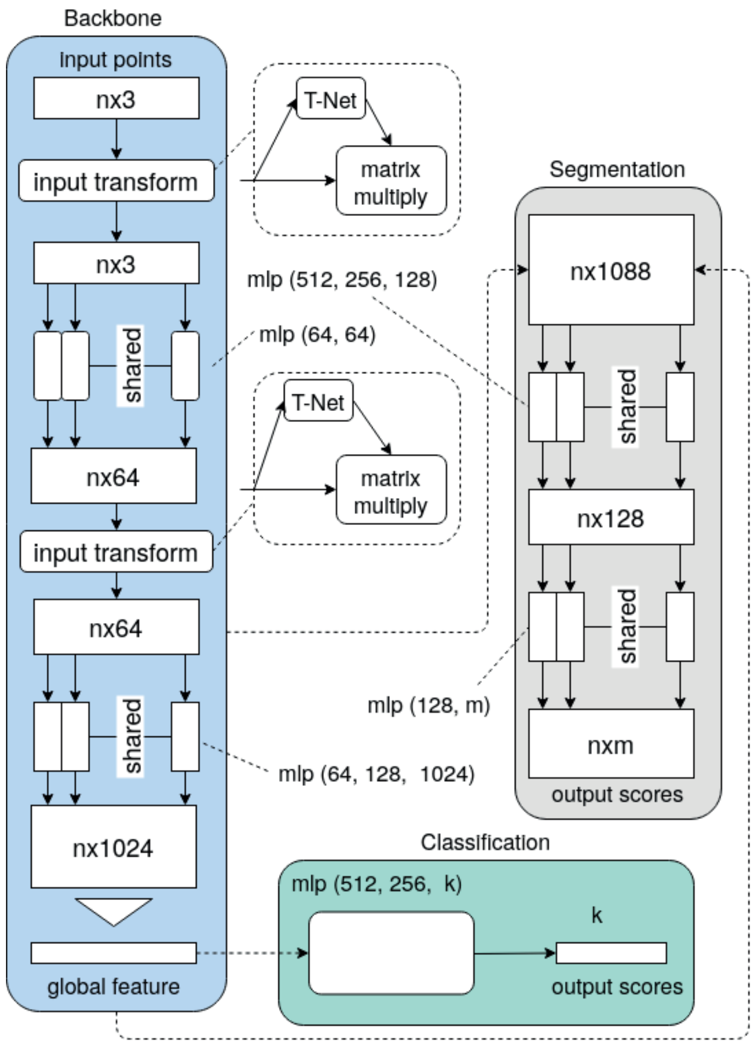


Figure 2.10: PointNet architecture [98] for a number of 3D recognition tasks. “mlp” stands for multi-layer perceptron, numbers in bracket are layer sizes. The T-net aims to learn an affine transformation matrix. The classification network takes n points as input and output classification scores for k classes. For segmentation network, the output after the second transformation network and output of the max pooling are concatenated for each point to extract new per point features.

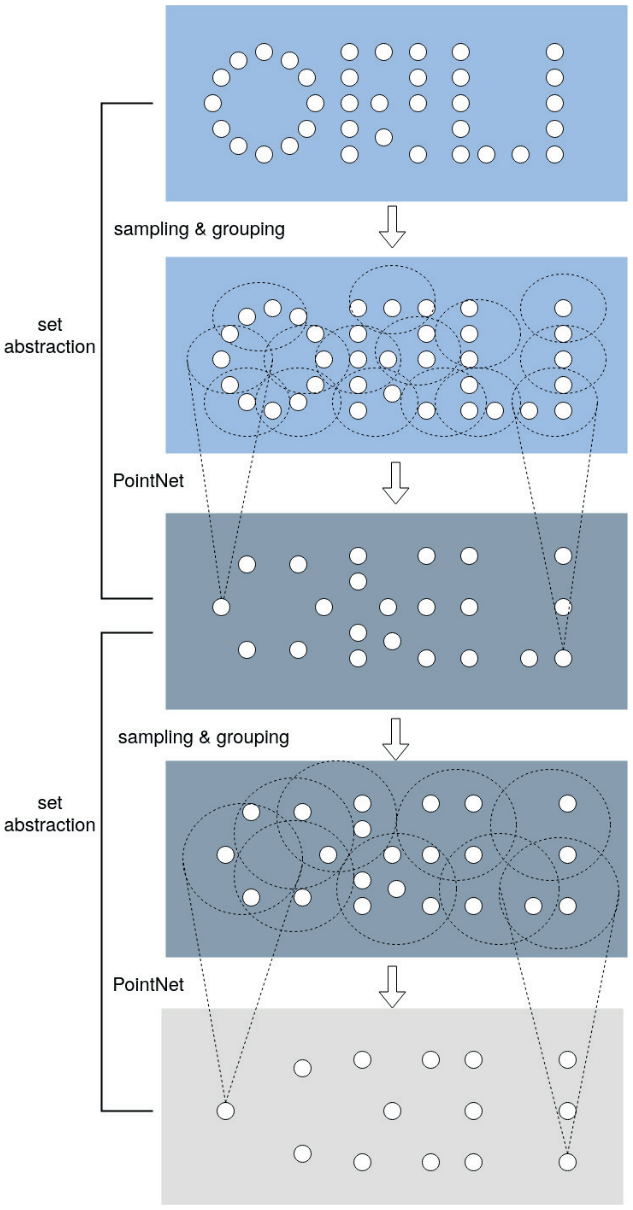


Figure 2.11: Illustration of feature learning architecture in PointNet++. The key element of PointNet++ network is set abstraction modules. Each module consists of a sampling layer, grouping layer, and PointNet layer.

2.4 Object Pose Estimation

2.4.1 Problem Statement

Given input data acquired by a sensor such as a monocular camera, RGB-D sensor, or LiDAR, we are interested in detecting objects and estimating their orientations and translations in 3D space. The problem is akin to 3D object detection, where the goal is to estimate oriented 3D bounding boxes of physical objects from sensor data as shown in Fig 2.12. The box bounds the complete object and is parameterized by its size (height, width, length), center, and orientation. However, the information from bounding boxes is often not applicable to robot manipulation. For example, in Fig 2.12 a mobile robot with forks is approaching pallets, but with the estimated bounding boxes it is still not able to find the right path to pick pallets up. Rather than detect objects with bounding boxes, aligning a full 3D model of the target object to its incomplete measured data would be more useful for manipulation tasks. To this end, we assume that the 3D models of objects are available and the object coordinate system $\mathcal{O}$ is defined in the 3D space of the model as illustrated in Fig. 2.13a. Object pose is represented by a rigid transformation from the object coordinate system to a reference coordinate system $\mathcal{G}$ (often camera coordinate system). The rigid transformation consists of a rotation $R \in SO(3)$ and a translation $t \in \mathbb{R}^3$, $\xi = [R|t]$. With an estimated object pose, we can perform the model-to-scene alignment as shown in Fig. 2.13.

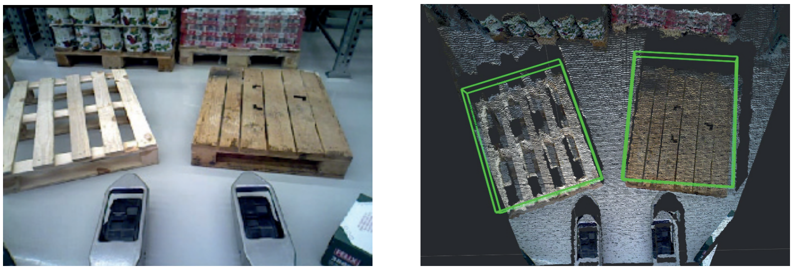

(a) (b)

Figure 2.12: Example of 3D object detection. (a) a scene in warehouse environment; (b) 3D point cloud from a RGB-D camera and estimated bounding boxes.

2.4.2 Conventional Methods

Conventional methods are mainly based on the matching of hand-crafted local or global features to extract the correspondence between observations (2D

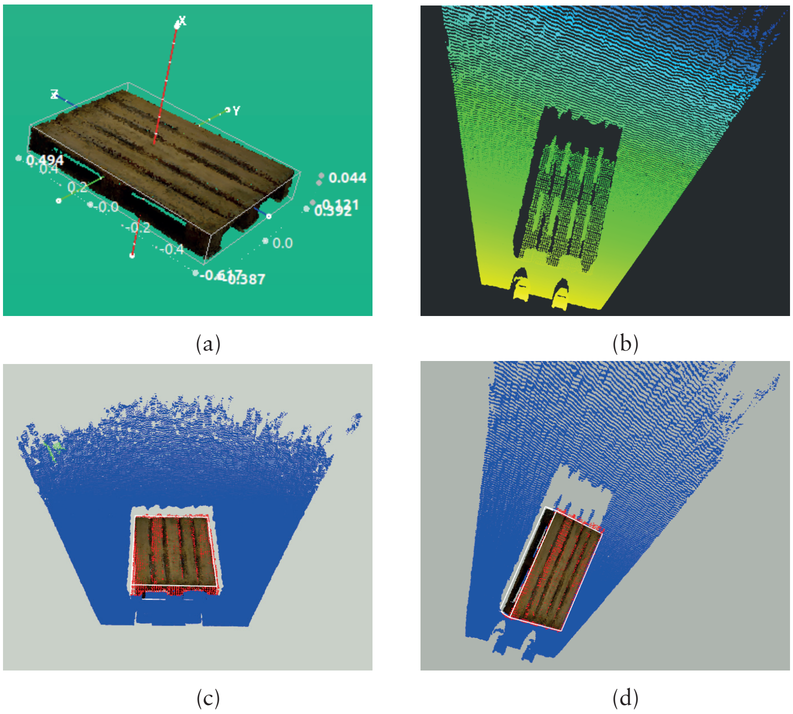


Figure 2.13: Example of 6D object pose estimation. (a) 3D model of a pallet; (b) 3D point cloud from a RGB-D camera; (c-d) the 3D model of pallet is aligned with the object cloud

images or 3D point clouds) and object mesh models. Here, the term “hand-crafted” refers to features manually designed in traditional methods which are distinct from learned features in deep neural networks. Global feature-based approaches utilize the whole geometric appearance of the object surface to define a single feature vector that effectively and concisely describes the entire 3D object [105, 131, 2]. On the contrary, the local feature-based methods exploit the geometric properties around specific keypoints [40]. While global methods are able to handle objects with self-similar surface parts, such as planar patches, spheres and cylinders, local approaches are more suitable for detecting and estimating the pose of complex objects in cluttered scenes. As a compromise solution, Drost et al. [26] build a global model description using point pair feature (PPF) and match that model locally using a fast voting scheme. The PPF method and its variants are well studied and commonly used as baseline methods in many previous works, including our work in Chapter 5. The following gives details about PPF and its variants.

The point pair feature (PPF) describes the relative position and orientation of two oriented points. More specifically, we denote $s_i \in S$ for points in the scene point cloud and $m_i \in M$ for points in the model point cloud. For two points m_1 and m_2 with normals n_1 and n_2, we set $d = m_2 - m_1$ and then the point pair feature F is defined as:

$$F(m_1, m_2) = (\| d \|_2, \measuredangle(n_1, d), \measuredangle(n_2, d), \measuredangle(n_1, n_2)) \tag{2.15}$$

where $\measuredangle(a, b) \in [0; \pi]$ denotes the angle between two vectors. The feature F is then used to build a global model description which is a mapping from the point pair feature space to the model. Fig. 2.14 shows an example of point pairs and their features on a single object. A hash table is used to represent model description and can be used to search matches between model features and scene features. While the global model description is accomplished in the offline phase, scene point pair features are calculated in the online phase. The potential matches then vote for an object pose via an efficient voting scheme and return the optimal object pose. The algorithm shows its high recognition performance and is utilized in a vast number of industrial and robotic applications. Since introduced, PPF has been improved and extended in many other works [16, 7, 49, 122, 39, 70]. Because of relying on a pair of points on the object surface and their normals, PPF is not discriminative enough for industrial parts which do not have rich variations in surface normals. To this end, [16] proposed several novel pair features that exploit boundary points and boundary line segments. The first new point pair feature is F_{B2B} (Boundary-to-Boundary) based on two points on the object boundary (depth edges). Instead of employing surface normals as in the original PPF, we fit line segments to boundary points and use their directions as orientations. We define B2B feature descriptor $F_{B2B} \in \mathbb{R}^4$ as

$$F_{B2B} = (\| d \|_2, \measuredangle(n_1^b, d), \measuredangle(n_2^b, d), \measuredangle(n_1^b, n_2^b)) \tag{2.16}$$

This descriptor is equivalent to the original PPF as in Eq. 2.15 except that n_1^b and n_2^b are directions of the 3D lines. The second new point pair feature is F_{S2B} (Surface-to-Boundary) based on an oriented surface point and an oriented boundary point. we define S2B feature descriptor $F_{S2B} \in \mathbb{R}^4$ as

$$F_{S2B} = (\| d \|_2, \measuredangle(n_1, d), \measuredangle(n_2^b, d), \measuredangle(n_1, n_2^b)) \tag{2.17}$$

Another point pair feature is F_{L2L} (Line-to-Line) using two 3D line segments. We define L2L feature descriptor $F_{L2L} \in \mathbb{R}^3$ as

$$F_{L2L} = (\| c_i - c_r \|_2, \measuredangle(l_r^2 - l_r^1, l_i^2 - l_i^1), d_{max}) \tag{2.18}$$

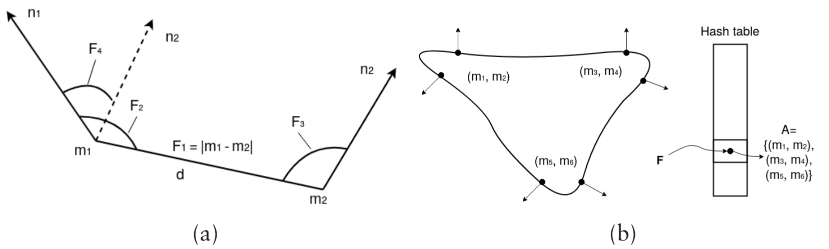


Figure 2.14: An example of buiding a global model descriptor using point pair feature (PPF) [26]: (a) Point pair feature $F = (F_1, F_2, F_3, F_4)$ of two oriented points; (b) The global model description.

where c_r and c_i are the closest points to lines that contain the 3D line segments, and $l_r^1, l_r^2, l_i^1, l_i^1$ are the end points of line segments. d_{max} is the maximum distance between two end points in two different segments. The experimental results shows that the pair features based on the object boundary improve the performance of object pose estimation for a wide class of industrial parts. While all points in the original PPF method contribute to the global feature descriptor equally, [7] argues that the points are not equally important to matching. Therefore, they perform weighting strategy based on the visibility per each vertex of the object surface. In addition, [7] introduced a coarse-to-fine segmentation step, a fast ranking and verification postprocessing steps to address the issues regarding the high dimensionality of the search space, sensitivity of the correspondence and the effect of outliers and low density surfaces. Although being extensively studied, PPF-based approaches share some common problems: (1) relying on searching a large set of paired feature correspondences severely limits their speed; (2) they are sensitive to measurement noise, heavy occlusion and background clutter.

2.4.3 Deep Learing-based Methods

Deep networks, especially Convolutional Neural Networks (CNN), have emerged as a powerful strategy for learning feature representations directly from RGB images and have led to significant progress in fundamental computer vision tasks such as image classification, object detection or semantic segmentation. Inspired by this success, 6D object pose estimation from a single RGB image using deep learning has been actively studied in recent years. Some works address the challenge by training CNN models to predict 2D keypoints and then compute 6D pose parameters with Perspective-n-Point (PnP) algorithms [93, 94, 100, 115]. Instead of detecting keypoints to serve as an intermediate representation for pose estimation, newer methods such as PoseCNN [133] directly estimate 6D poses from image data. The PoseCNN architecture employs semantic labeling which provides richer information about the objects. PoseCNN recovers the 3D translation of an object by localizing its center in the image and estimating the 3D center distance from the camera. The 3D rotation of the object is estimated by regressing convolutional features to a quaternion representation. In addition, in order to handle symmetric objects, the authors introduce ShapeMatch-Loss, a new loss function that focuses on matching the 3D shape of an object. The results show that this loss function produces superior estimation for objects with shape symmetries. However, this approach requires Iterative Closest Point (ICP) for refinement which is prohibitively slow for real-time applications.

Although deep learning-based methods significantly improve the pose estimation accuracy compared to the traditional methods, features learned from only RGB images are sensitive to occlusion and illumination changes, which places them far from being applied to complicated scenes. Moreover, the methods mainly rely on 2D images to estimate 6D pose of objects, which could lead to losing geometric constraint information of rigid objects. When depth information is available, it can be combined with RGB images to provide a richer representation and improve the performance of object pose estimation. Wang et al. proposed DenseFusion [121] which is approximately 200x faster than PoseCNN-ICP and outperforms previous approaches on two datasets, YCB-Video and LineMOD. The key technique of DenseFusion is that it extracts features from the color and depth images and fuses RGB values and point clouds at the per-pixel level. This per-pixel fusion scheme enables the model to reason about the local appearance and geometry information, which is essential to handle occlusions between objects. In addition, an end-to-end iterative pose refinement procedure is proposed to further improve pose estimation while achieving near real-time inference.

Fig. 2.15 illustrates the overall architecture of DenseFusion. It is composed of three components: a) a PointNet-like network that processes each point in the masked 3D point cloud to a geometric feature embedding, b) a CNN-based image embedding network that processes the color information to extract per-pixel features representing the appearance information of the input image at the corresponding location, c) a pixel-wise fusion network that combines both embeddings, performs local per-pixel fusion and makes pose predictions based on each fused feature. The core of DenseFusion is the fusion network that embeds and fuses the appearance and geometric information at the per-pixel level. This fusion scheme enables the model to potentially favor the predictions based on the seen part of the object and soften the effects of occlusion and measurement noise. The loss is defined as the distance between the points sampled on the object model in ground truth pose and corresponding points on the same model transformed by the estimated pose:

$$L_i^p = \frac{1}{M} \sum_j \| (Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i) \| \tag{2.19}$$

where x_j denotes the j^{th} point of the M randomly selected 3D points from the object's 3D model, $\xi = [R|t]$ is the ground truth pose, and $\hat{\xi}_i = [\hat{R}_i|\hat{t}_i]$ is the predicted pose generated from the fused embedding of the i_{th} dense-pixel. For objects with symmetric views, the metric is adapted by computing the average distance using the closest point distance:

$$L_i^p = \frac{1}{M} \sum_j \min_{0<k<M} \| (Rx_j + t) - (\hat{R}_i x_k + \hat{t}_i) \| \tag{2.20}$$

Although DenseFusion has achieved promising results, like other single-view-based methods it suffers significantly from the ambiguity of object appearance and occlusions in cluttered scenes, which are very common in practice. In addition, since DenseFusion relies on segmentation results for pose prediction, its accuracy highly depends on the performance of the segmentation framework used. As in pose estimation networks, if the input to a segmentation network contains an occluder, the occlusion significantly influences the network output. Not only DenseFusion but also most of the current CNN-based methods consider objects independently and estimate their poses using a single input (RGB or RGB-D) image. However, scenes are often composed of different object instances and multiple images of the scene can easily be captured by a single moving camera, or in a multiple-camera setup. In Chapter 4, while exploiting the advantages of DenseFusion framework, we address the problem of the ambiguity of object appearance and occlusion by combining information from multiple views and estimates jointly the pose of multiple objects to obtain a single consistent scene interpretation.

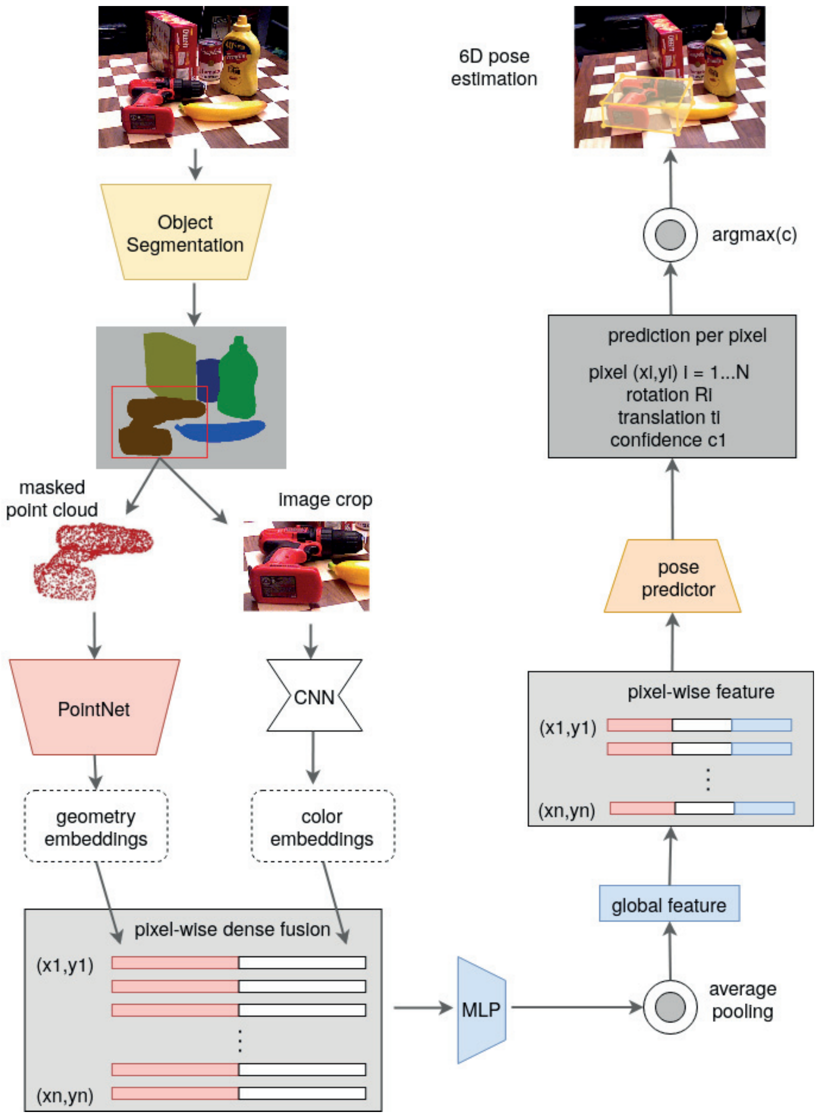


Figure 2.15: Architecture of DenseFusion [121] for 6D object pose estimation. Color images are processed by a segmentation framework to generate object masks and bounding boxes. Then, for each segmented object, the RGB colors and point cloud from the depth map are encoded into embeddings and fused at each corresponding pixel. The pose predictor produces a pose estimate for each pixel and the predictions are voted to generate the final 6D pose prediction of the object.

Chapter 3
Registration of RGB-D Images

3.1 Introduction

The ability to quickly acquire a 3D map (model) of an unknown environment is of extreme importance in robot manipulation. It enables robots to operate safely and effectively in applications with a high demand on flexibility. For instance, automated picking and manipulation of boxes and other types of goods in warehouses requires an accurate model of the world to help robots better understand their surroundings, and to, for example, avoid undesirable contacts with the environment. With the increasing availability of RGB-D sensors, research on 3D mapping has made giant strides in development [129, 87, 111, 127, 13, 23]. These approaches achieve dense surface reconstruction of complex scenes while maintaining real-time performance through implementations on highly parallelized hardware. Most of these approaches have a very similar processing pipeline. In the first stage, noise reduction and outlier removal are applied to the raw depth measurements and then 3D points are generated. Additional information such as normals also might be extracted from the depth image. In the next step, the current camera pose relative to the scene is estimated via a registration algorithm in a frame-to-frame or frame-to-model fashion by minimizing a cost function. Finally, the surface measurements are fused into an accumulated global model based on the camera pose determined in the previous stage. Camera pose estimation is the crux to 3D reconstruction systems and its performance mainly depends on the registration process. Therefore, in this chapter, we focus on registration algorithms.

A large number of registration algorithms have been proposed in the context of RGB-D Tracking and Mapping (TAM) [129, 87, 57, 28]. Feature-based approaches estimate the sensor pose by only considering informative and characteristic points known as key points [57, 28]. Alternatively, dense geometric tracking approaches, such as KinectFusion [87], typically apply an ICP [15] variant to directly register the full depth image to an online reconstructed volumetric model. The original KinectFusion algorithm uses a Truncated Signed

Distance Function (TSDF) for model representation and point-to-plane ICP [15] for alignment. Several alternatives to this choice of algorithms have been proposed [85, 13], which are expected to perform better in regions where the point-to-plane distance is ill-defined. Using only depth data, tracking failure can occur in situations where the amount of characteristic features in the depth map is low. Steinbrucker et al. [111] introduced an energy minimization approach for RGB-D image registration that relies on color information instead. In comparison with geometric ICP, the authors reported that their method is more accurate in the regime of small camera motions. Whelan et al. [127] combined the color and depth information in the cost function so that all given information is used. They demonstrated that this combination increases the robustness of camera tracking across a variety of environments. Besides depth and color images, semantic information is now often available due to the improvement of semantic image segmentation driven by deep learning. However, there is still a lack of studies that examine the use of semantic cues in improving the performance of the registration process.

In this chapter, we review current registration algorithms and explore the benefits of utilizing segmentation in camera tracking. We propose modifications of the registration objective function to make full use of the semantic cues in the process. For a fair comparison, we employ the pipeline of ElasticFusion as a backbone for all registration methods and evaluate them within this framework. The rest of this chapter is organized as follows. Section 3.2 provides preliminaries, while a problem statement is formalized in Section 3.3. Section 3.4 presents a discussion on existing registration methods. Having discussed current registration algorithms, Section 3.5 then shifts the focus of this chapter to our proposed approach for reliable camera pose tracking. The results are then presented and analyzed in Section 3.6. Finally, Section 3.7 summarizes the main contributions and lessons learned in this chapter.

3.2 Preliminaries

Following ElasticFusion [129], the reconstruction system maintains a fused surfel-based model of the environment. The model is represented by a cloud of surfels $\mathcal{M}^s$, where each surfel consists of a position (vertex) $v \in \mathbb{R}^3$, normal $n \in \mathbb{R}^3$, color $c \in \mathbb{N}^3$, initialization timestamp t_0 and last updated timestamp t. The image space domain is defined as $\Omega \subset \mathbb{N}^2$, where an RGB-D frame is composed of a RGB image I_{rgb} and a depth image I_d of depth pixels:

$$I_{rgb} : \Omega \subset \mathbb{N}^2 \to \mathbb{N}_3 \tag{3.1}$$

$$I_d : \Omega \subset \mathbb{N}^2 \to \mathbb{R} \tag{3.2}$$

We define the 3D back projection of a point $p \in \Omega$ given a depth image I_d as

$$v(p) = [\frac{p_x - c_x}{f_x} I_d(p), \frac{p_y - c_y}{f_y} I_d(p), I_d(p)]^\intercal \tag{3.3}$$

Considering the camera intrinsics matrix

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

and the perspective projection of a 3D point

$$\Pi([x, y, z]^\intercal) = [x/z, y/z]^\intercal \tag{3.5}$$

we can express $v(p)$ and $p(v)$ as

$$v(p) = K^{-1} I_d(p)(p, 1)^\intercal \tag{3.6}$$
$$p(v) = \Pi(Kv) \tag{3.7}$$

Given a RGB image I_{rgb} with color $c(p) = [c_1, c_2, c_3]^\top$, the intensity value of a pixel $p \in \Omega$ is defined as $I_{rgb}(p) = (c_1 + c_2 + c_3)/3$.

3.3 Problem Statement

Given a new RGB-D frame $[I_{rgb}^{t+1}, I_d^{t+1}]$ at time step $t + 1$, we wish to find the global pose of the camera T^{t+1} (*w.r.t.* a global frame $\mathcal{F}_G$). We represent the 6DOF camera pose estimated by a rigid body transformation matrix:

$$T^{t+1} = \begin{bmatrix} \multicolumn{3}{c}{\mathbf{R}^{t+1}} & \mathbf{t}^{t+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \tag{3.8}$$

consisting of a rotation $\mathbf{R}^{t+1} \in SO(3)$, parametrized as a matrix $\in \mathbb{R}^{3\times 3}$ and a translation $\mathbf{t}^{t+1} \in \mathbb{R}^3$. Since we already know the transformation from the previous time step T^t, we only need to find the incremental transformation ΔT between t and $t + 1$. Then T^{t+1} can be computed as

$$T^{t+1} = T^t \Delta T \tag{3.9}$$

So we first move the new measurements by ΔT such that they are aligned with the previous data in their respective coordinate system and then transform them to global coordinates using T^t. To find ΔT, we need to align the two consecutive RGB-D image pairs $[I_{rgb}^{t+1}, I_d^{t+1}]$ and $[I_{rgb}^t, I_d^t]$ to get our desired result. This alignment is called frame-to-frame tracking. However, in many systems, they instead track the new camera frame by aligning $[I_{rgb}^{t+1}, I_d^{t+1}]$ against the model prediction $[\hat{I}_{rgb}^t, \hat{I}_d^t]$ from the previous frame, called frame-to-model. A successful alignment will give us the incremental transformation ΔT. Note that we assume the camera has a small motion and the scene remains static.

3.4 Prior Methods

3.4.1 Geometric Pose Estimation

Many of the previous works rely on the iterative closest point (ICP) [15, 85, 13] or its variants to align depth images for camera pose tracking [87, 128, 13, 85]. The general idea is to iteratively assign correspondences between the points of the two frames and to register them. Newcombe et al. [87] introduced the first system (KinecFusion) which permits real-time and dense 3D surface reconstruction using an RGB-D camera through extensive GPGPU parallelization. The original registration algorithm used in KinectFusion is a variant of the ICP algorithm and performs data alignment in a frame-to-model fashion. It estimates the sensor pose for each new frame by utilizing a point-to-plane error metric in which the objective of minimization is the sum of the squared distance between a point from a live surface measurement and the tangent plane at its correspondence point from the model prediction. The cost function performs well in environments with high geometric textures.

Given the predicted depth image $\hat{I}_d^t$ from the last frame at time step t, the back-projection of the $i-th$ vertex in $\hat{I}_d^t$ is defined as:

$$\hat{v}_i^t := \hat{v}^t(p_i) \tag{3.10}$$

In the setting, the vertices obtained by back-projection of the new depth image I_d^{t+1} describe the scene in the local camera coordinate system at time t:

$$v_i^{t+1} := \Delta T v^{t+1}(p_i) \tag{3.11}$$

We assume that there exists a set of correspondence pairs between the two points sets. KinectFusion exploits the fact that these points are generated from a camera with known intrinsic parameters. Computing the correspondences is therefore done by the fast projective data association algorithm [8]. For each vertex $v^{t+1}(p_i)$, which is defined in the local coordinate system at time $t+1$, we compute the vertex $\hat{v}_{c(i)}$ at time t that is at the same line of sight in the local system at time t. This means, we first need to know the coordinates of vertex

$v_{t+1}(p_i)$ in the other coordinate system by applying ΔT, then project it to pixel coordinates and use this pixel $p_{c(i)}$ as a look up for the corresponding vertex:

$$\hat{v}^t_{c(i)} = v^t(p(\Delta T v^{t+1}(p_i))) \tag{3.12}$$

After having calculated the correspondences, a measure of error as the sum of squared distances between corresponding points in 3D can be computed:

$$E_{geo} = \sum_i \parallel (v_i^{t+1} - \hat{v}^t_{c(i)}) \cdot \hat{n}^t_{c(i)} \parallel^2 \tag{3.13}$$

This formulation is the point-to-plane error function as described in [87]. In particular, we wish to find a rigid transformation ΔT that minimizes the error between the correspondences:

$$\min_T . \sum_i \parallel (v_i^{t+1} - \hat{v}^t_{c(i)}) \cdot \hat{n}^t_{c(i)} \parallel^2 \tag{3.14}$$

$$\xi \to \Delta T = \exp(\hat{\xi})\Delta T \tag{3.15}$$

The problem is solved by an iterative estimation process and ΔT is updated in every iteration via the estimated motion parameters $\xi = [\omega^T \; x^T]^T$, $\omega \in \mathbb{R}^3$ and $x \in \mathbb{R}^3$. The hat operator forms the matrix:

$$\hat{\xi} = \begin{bmatrix} & [\omega]_\times & & \mathbf{x} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.16}$$

where $[\omega]_\times \in \mathbb{R}^{3\times3}$ is a skew-symmetric rotation component:

$$\omega]_\times = \begin{bmatrix} 0 & -\omega_2 & \omega_1 \\ \omega_2 & 0 & -\omega_0 \\ -\omega_1 & \omega_0 & 0 \end{bmatrix} \tag{3.17}$$

We can re-write the error function as:

$$E_{icp} = \sum_i \parallel (\exp(\hat{\xi}) v_i^{t+1} - \hat{v}^t_{c(i)}) \cdot \hat{n}^t_{c(i)} \parallel^2 \tag{3.18}$$

We assume that motions are small, then we have:

$$\exp(\hat{\xi}) = \sum_0^\infty \frac{\hat{\xi}^n}{n!} \approx I + \hat{\xi} \tag{3.19}$$

By plugging this approximation into the error function and rearranging the terms as described in [127], then we get the final form of energy equation:

$$E_{icp} = \sum_i \| \begin{bmatrix} v_i^{t+1} \times \hat{n}_{c(i)}^t \\ \hat{n}_{c(i)}^t \end{bmatrix} \xi + (v_i^{t+1} - \hat{v}_{c(i)}^t) \cdot \hat{n}_{c(i)}^t \|^2 \tag{3.20}$$

$$= \| J_{geo}\xi + r_{geo} \|^2 \tag{3.21}$$

To estimate the motion update vector ξ, we compute the least-squares solution:

$$\underset{\xi}{\operatorname{argmin}} \| J_{geo}\xi + r_{geo} \|^2 \tag{3.22}$$

3.4.2 Photometric Pose Estimation

In Section 3.4.1 above, the dense point-plane based ICP minimizes the cost over the error E_{icp} between vertices in the current depth frame and the predicted depth image. The cost function performs well in environments with high geometric texture, however tracking failures can occur in case there are not enough features to fully constrain all 6DOF of the camera pose. For instance, if the measured points are located on planar surfaces then the point-to-plane error metric will fail to register successive views. This is because there will be no mechanism to guarantee that a global minimum can be reached by shifting source points to target points in the direction perpendicular to the normals. Steinbrucker et al. [111] used appearance information to overcome this. In addition to the assumption of small camera motion and static scene, we also can assume that every surface point has the same intensity in all images. This means the appearance of the scene is the same in both images. Based on this assumption, we aim to find the motion parameters ξ that minimize the cost over the photometric error (intensity difference) between pixels:

$$E_{rgb} = \sum_i \| I_{rgb}^{t+1}(p_i) - \hat{I}_{rgb}^t(\Psi(\xi, p_i)) \|^2 \tag{3.23}$$

$$\Psi(\xi, p_i) = \Pi(K\xi\Delta T v^{t+1}(p_i)) \tag{3.24}$$

According to the original algorithm by Steinbruecker et al. [111], in a similar manner as E_{icp} we get J_{rgb} and r_{rgb} for the color (intensity) correspondences. Then we have the final form of energy equation

$$E_{rgb} = \| J_{rgb}\xi + r_{rgb} \|^2 \tag{3.25}$$

And similar to the geometric pose estimation method, we solve least-squares problem to estimate the motion parameters ξ.

3.4.3 Joint Optimization

Using only depth information, tracking failure can occur in situations where the scene has no distinctive shapes such as a large planar scene or spherical surface. Similarly, a reliance on color information alone for the estimation of the camera pose suffers from the lack of visual features and does not exploit the rich geometric information contained within the scene. As a result, Whelan et al. [130, 129] uses both dense depth and photometric information. They combine the cost functions of both the geometric and photometric estimates in a weighted sum. The sum of the RGB-D and ICP cost is defined as

$$E_{icp\_rgb} = E_{icp} + \omega_{rgb} E_{rgb} \tag{3.26}$$

where ω_{rgb} is the weight and was set empirically to 0.1 to reflect the difference in metrics used for E_{icp} and E_{rgb} costs. To optimize this cost, Whelan et al. [127, 130] use the Gauss-Newton non-linear least-squares method and exploit the GPU to accelerate the minimization process. The details of implementation can be found in [130, 129].

3.5 Joint Optimization with Semantic Information

While the ICP method does not work well in scenarios with repetitive geometric structures, the largest drawback of photometric pose estimation is their susceptibility to illumination and changes in camera exposure time. On the contrary, semantic segmentation label image offers the favorable property of being largely invariant to both environmental conditions as well as the surface characteristic. In this section, we propose modifications of the registration cost function to make full use of the semantic class labels in the process. The proposed objective function features tunable weights for the depth, appearance, and semantic information channels, which are learned from data. A fast semantic segmentation and registration weight prediction convolutional neural network (Fast-RGBD-SSWP) suited to efficient computation is introduced.

3.5.1 Sematic Segmentation

Our segmentation framework for Fast RGBD Semantic Segmentation and Weight Prediction (Fast-RGBD-SSWP) is inspired by Fast-SCNN [96] and FuseNet [43] to address the problem of real-time semantic labeling on RGB-D data. We employ depthwise separable convolutions and residual bottleneck blocks for deep CNNs [106]. The network contains two branches to extract features from RGB and depth images, and the depth feature map is constantly fused into the RGB branch as shown in Fig. 3.1. In each branch, only three layers are employed to extract low-level features for the purpose of feature sharing. The first layer is a standard convolutional layer (Conv2D) and the remaining two layers are depthwise separable convolutional layers (DSConv) [106].

The low-level features not only become the input for the other stages of semantic segmentation but also share computation with the branches for per-image adaptive weight estimation. The weight prediction is treated as a classification problem where the target is a binary decision whether or not the given RGB image and depth image should be used in the registration process. In other words, we aim to train our weight predicting model as a binary classifier, where one class signifies that the image contains useful information for the subsequent registration process, while the other class indicates the converse. The probability predicted from the classification model is considered as an adaptive weight for our joint cost function for camera pose estimation.

Similar to Fast-SCNN, the semantic segmentation branch includes a global feature extractor, a feature fusion module and a standard classifier as shown in Fig. 3.1. However, instead of using feature maps from the RGB branch, our global feature extractor module takes the feature maps fused by the depth and RGB branches. This module is composed of efficient bottleneck residual blocks [106] and a pyramid pooling module (PPM) [141]. The bottleneck block uses depthwise separable convolution to enhance efficiency without significantly reducing effectiveness. The feature fusion module processes a simple addition of features as utilized in ICNet [140]. In the classifier, two depthwise separable convolutions (DSConv) and one pointwise convolution (Conv2D) are employed. Softmax is used during training and inference. The output of the CNN is a per-pixel independent probability distribution over the class labels $P(l_i)(u)$, $l_i \in \mathcal{L}$ with u denoting pixel coordinates. $\mathcal{L}$ is a predetermined set of L semantic classes encoded by $\mathcal{L} := \{0, ..., L-1\}$.

3.5.2 Joint Optimization

Rather than rely on only geometric and photometric data, we additionally employ semantic information to perform registration. The cost we wish to minimize depends on the difference in predicted likelihood values between the label probability maps:

$$E_{sem\_full} = \sum_i \sum_j \| P(l_j)(p_i) - P(l_j)(\Psi(\hat{\xi}, p_i)) \|^2 \tag{3.27}$$

$$\Psi(\xi, p_i) = \Pi(K\xi\Delta T v^{t+1}(p_i)) \tag{3.28}$$

To simplify minimizing the cost function, we only take the probability of the most likely class on each pixel-wise probability vector $Q(u, P) = \max P(l_i)$ from frame t and the probability of the same class label from frame $t+1$. We denote values of $Q(u, P)$ over a given image as a semantic probability map. So

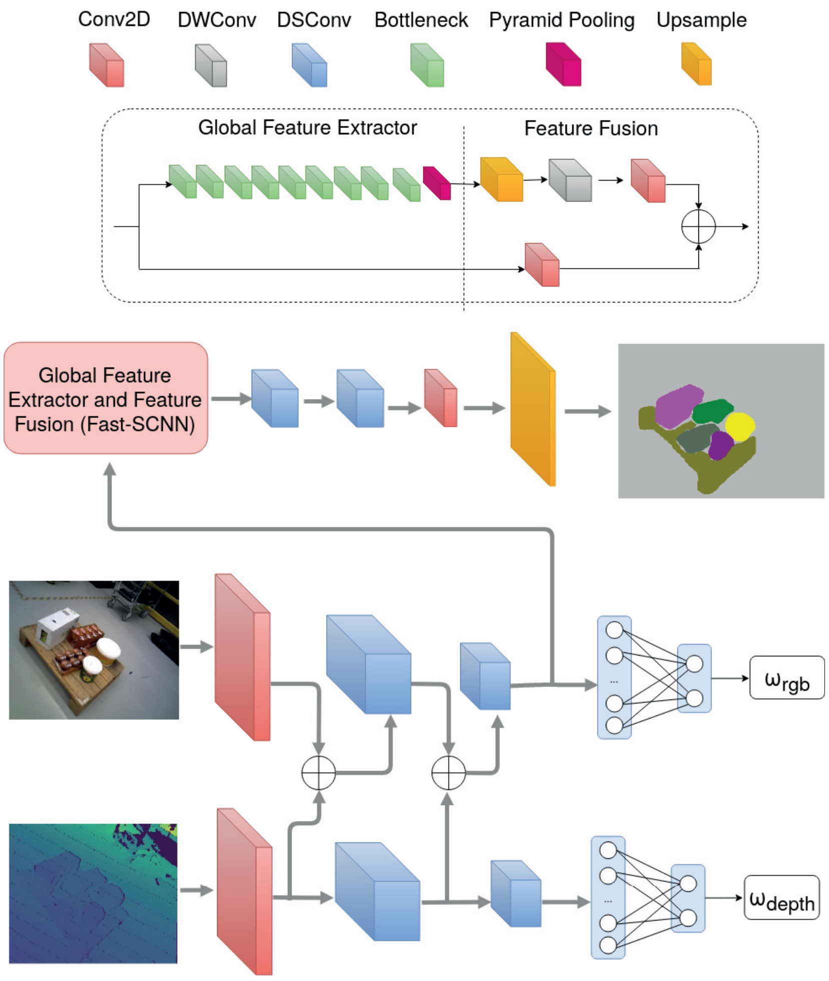


Figure 3.1: Fast-RGBD-SSWP makes dense predictions inferring labels for every pixel while simultaneously yielding per-image adaptive weights for camera tracking. The network uses standard convolution (Conv2D), depth-wise separable convolution (DSConv), depth-wise convolution (DWConv), inverted residual bottle-neck blocks (bottleneck), a pyramid pooling module and a feature fusion module block.

based on this simplification, the semantic probability error can be formulated as:

$$E_{sem} = \sum_{i} \| Q(p_i, P_{t+1}) - Q(\Psi(\hat{\xi}, p_i), P_t) \|^2 \tag{3.29}$$

In words, P_{t+1} and P_t are per-pixel independent probability distributions over the class labels from the frame at time step $t+1$ and t respectively. There are alternatives, and later discussion in the experimental results we will explain why we decided to go with the simplified function.

At this point, we aims to estimate a sensor pose that minimizes the cost over a combination of the global point-plane energy, photometric error, and semantic difference. We wish to minimize a joint optimization objective:

$$E_{icp\_rgb\_sem} = \omega_{geo} E_{icp} + \omega_{rgb} E_{rgb} + \omega_{sem} E_{sem} \tag{3.30}$$

where $\omega_{geo} E_{icp}$, $\omega_{rgb} E_{rgb}$, and $\omega_{sem} E_{sem}$ are the geometric, photometric and semantic error terms respectively. The geometric and photometric error functions are weighted by factors predicted from the Fast-RGBD-SSWP network. The weight for semantic error is defined as $\omega_{sem} = N_m / N_u$, where N_m is the number of non-background pixels and N_u is the number of pixels per frame. This fraction accurately captures the amount of semantic texture present in the scene. Finally, we find the transformation by minimizing the objective 3.30 through the Gauss-Newton non-linear least-square method as in the previous Section 3.4.3.

3.5.3 Per-pixel Adaptive Weights

In Eq. 3.30 we combine adaptively weighted photometric, geometric and semantic cost terms in a single objective function. These adaptive weights are chosen on a per-image basis, while ideally they should be different for each pixel, as certain regions in the image can contain varying amounts of structure and color. We use this observation and propose to estimate per-pixel adaptive weights based on textureness assessment (Fig. 3.2). To define the textureness of each depth image pixel, we assume that untextured regions are often piecewise flat and thus the amount of characteristic features is low. Under these assumptions, the idea behind our proposed cost function is to favor highly textured regions of the image. The energy is adaptively weighted based on the local variance at $u = p_i$, we define it as in [120]:

$$\lambda(u) = \frac{\sigma_u^2}{\sigma_u^2 + \epsilon} \tag{3.31}$$

where σ_u denotes the local variance of the 5x5 patch around pixel u in the current image and ϵ is an empirically set constant. The higher the variance, the

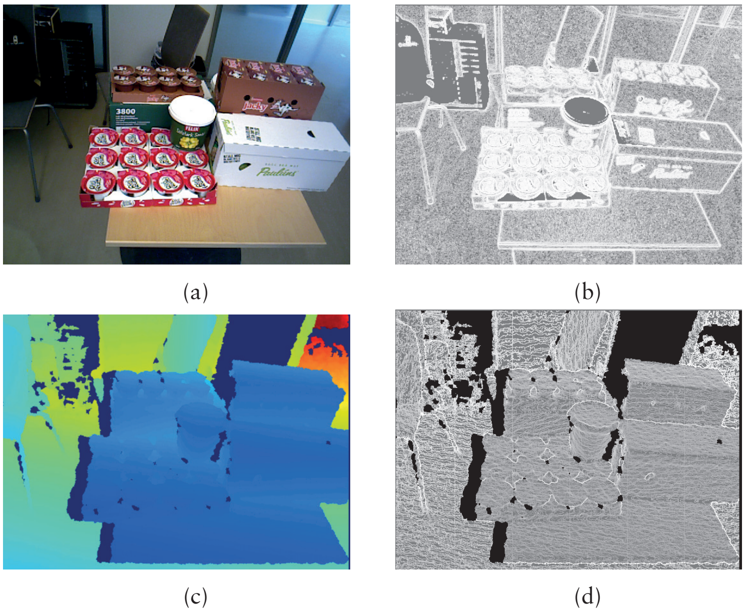


Figure 3.2: Visualization of per-pixel weights computed on color (a) and depth (c) images, higher weights are whiter in (b) and (c). (a) Color image; (b) Weights on color image; (c) Depth image; (d) Weights on depth image.

closer the weight is to 1. Fig. 3.2 shows an example of per-pixel weights for a RGB-D image. We can reformulate the geometric energy E_{icp}, the photometric energy E_{rgb}, and the joint objective $E_{icp\_rgb\_sem}$ with per-pixel adaptive weights as following:

$$E^{pa}_{icp} = \sum_i \lambda_{icp}(p_i) \parallel (\exp(\hat{\xi})v^{t+1}_i - \hat{v}^t_{c(i)}) \cdot \hat{n}^t_{c(i)} \parallel^2 \tag{3.32}$$

$$E^{pa}_{rgb} = \sum_i \lambda_{rgb}(p_i) \parallel I^{t+1}_{rgb}(p_i) - \hat{I}^t_{rgb}(\Psi(\xi, p_i)) \parallel^2 \tag{3.33}$$

$$E^{pa}_{icp\_rgb\_sem} = E^{pa}_{icp} + E^{pa}_{rgb} + \omega_{sem}E_{sem} \tag{3.34}$$

where the weights λ_{icp} and λ_{rgb} are computed from Eq. 3.31 with the varaince σ_u taken as the variance of a local 5x5 patch of pixels from the depth and intensity images.

3.6 Evaluation

In this section, we evaluate the presented registration algorithms through experiments on the standard TUM RGB-D dataset [112] and a newly collected warehouse dataset. Our experiments are aimed at evaluating trajectory estimation and surface reconstruction accuracy. For all tests, we ran our system on a desktop PC running 64-bit Ubuntu 18.04 Linux with an Intel(R) Xeon(R) E-2176G CPU 3.70GHz and an Nvidia GeForce RTX 2080 Ti 10GB GPU. For a fair comparison, we employ the pipeline of ElasticFusion as a backbone for all registration methods and evaluate them within this framework. The Fast-RGBD-SSWP network is implemented using PyTorch 1.0. To train Fast-RGBD-SSWP we used stochastic gradient descent (SGD) with momentum 0.9 and batch-size 12. In all of the presented experimental setups, results are generated from RGB-D video with a resolution of 640x480 pixels.

3.6.1 The Warehouse Object Dataset

Unlike scenes recorded in other publicly available datasets, warehouse environments pose more complex problems, including low illumination inside shelves, low texture and symmetric objects, clutter, and occlusions. To advance applications of robotics to warehouse logistics as well as to thoroughly evaluate our method, we collected an RGB-D video dataset containing instances of the 11 objects shown in Fig. 3.3. The dataset focuses on the challenges in reconstructing unknown environments and detecting warehouse object poses using an RGB-D sensor. The dataset consists of over 100,000 RGB-D images extracted from 100 videos captured by an ASUS Xtion PRO Live sensor, the 6D poses of the objects and ground truth instance segmentation masks manually generated using the LabelFusion framework [78], as well as camera trajectories from a motion capture system developed by Qualisys[1]. Calibration is required for both the RGB-D sensor and motion capture system shown in Fig. 3.4. We calibrated the motion capture system using the Qualisys Track Manager (QTM) software. For RGB-D camera calibration, the intrinsic camera parameters were estimated using the classical black-white chessboard and the OpenCV library. For extrinsic calibration, four markers were placed on the outer corners of the checkerboard as in [112]. We also attached four spherical markers on the sensor. Similar to [112], we were able to estimate the transformation between the pose from the motion capture system and the optical frame of the RGB-D camera. Fig. 3.5, Fig. 3.6 and Fig. 3.7 show examples of scenes from the warehouse dataset.

[1] `https://www.qualisys.com`

(a) Waffle

(b) Jacky

(c) Skansk

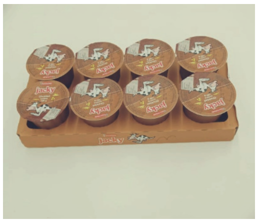
(d) Sotstark

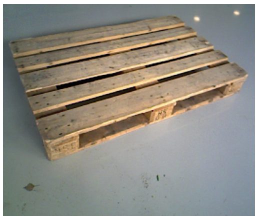
(e) Onos

(f) Risi Frutti

(g) Pauluns

(h) Tomatpure


(i) Small Jacky


(j) Pallet

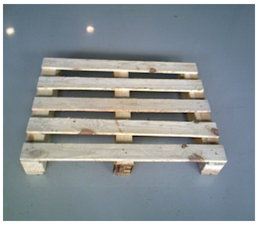
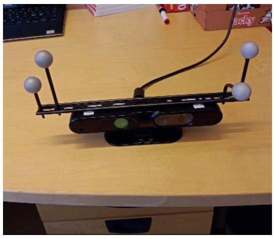
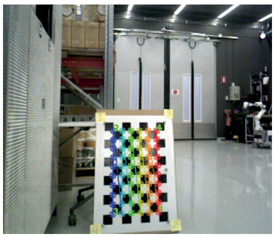
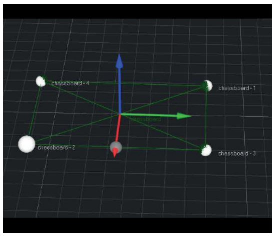

(k) Half Pallet

Figure 3.3: The set of 11 objects in the warehouse object dataset.

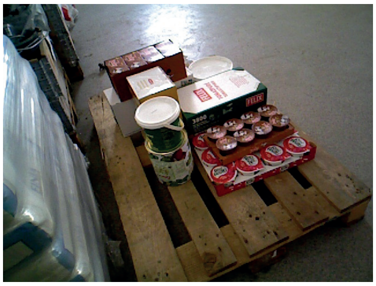

(a)

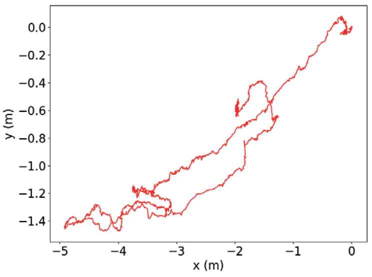

(b)



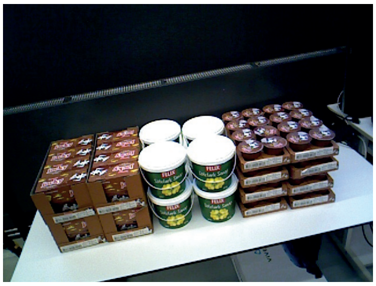
(c)

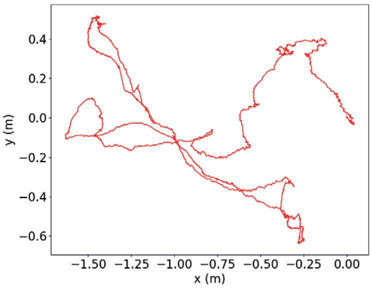
Figure 3.4: We collected a dataset for the evaluation of reconstruction and pose estimation systems in a typical warehouse using (a) a hand-held ASUS Xtion PRO LIVE sensor. Calibration parameters were found by using (b) a chessboard and (c) reflective markers detected by the motion capture system.


(a)

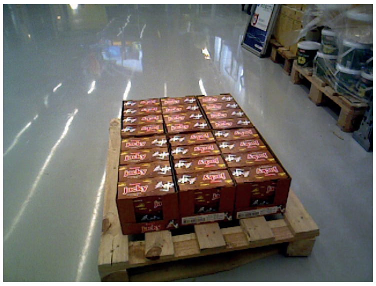


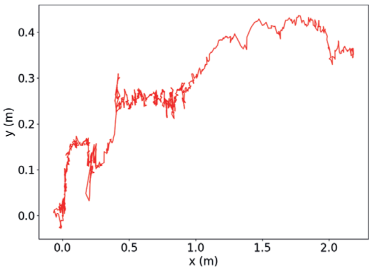
(b)


(c)

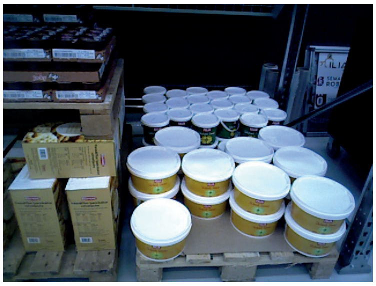


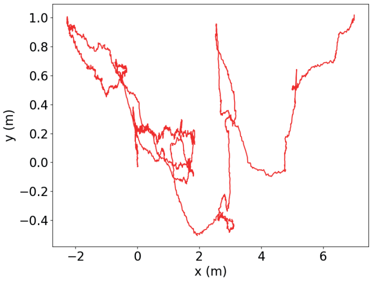
(d)


(e)



(f)

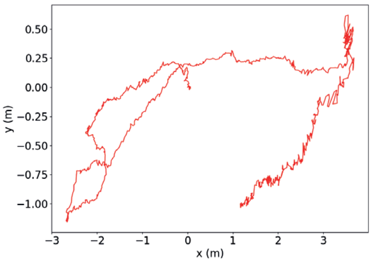

(g)



(h)

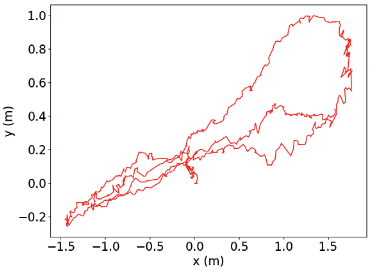
Figure 3.5: Sequences (a-b) warehouse_01, (c-d) warehouse_02, (e-f) warehouse_03, and (g-h) warehouse_04 in the warehouse dataset. (left) scenes; (right) camera trajectories.

(a)



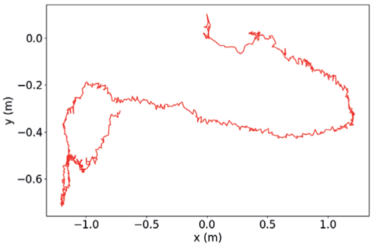
(b)

(c)

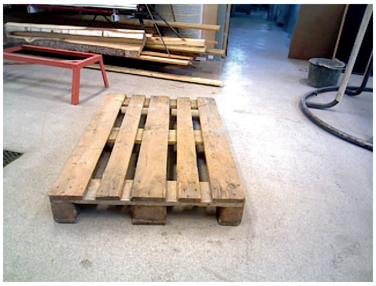


(d)

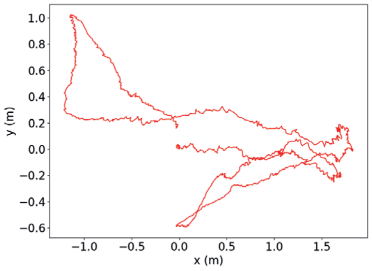
(e)



(f)



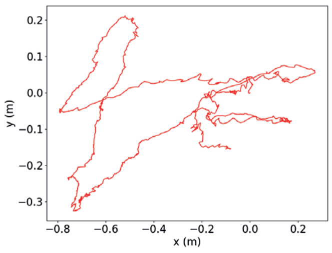
(g)



(h)

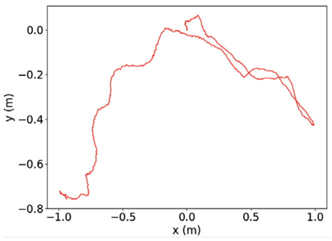
Figure 3.6: Sequences (a-b) warehouse_05, (c-d) warehouse_06, (e-f) warehouse_07, and (g-h) warehouse_8 in the warehouse dataset. (left) scenes; (right) camera trajectories.

(a) recorded scene



(b) camera trajectory

(c) recorded scene



(d) camera trajectory

Figure 3.7: Sequences warehouse_09 and warehouse_10.

3.6.2 Trajectory Estimation

We compare the trajectory estimation performance of the registration method using different error functions on the TUM RGB-D dataset and the warehouse dataset. The test set covers a large variety of scenes and camera motions and provides sequences for debugging with slow motions as well as longer trajectories with and without loop closures. Each sequence contains the color and depth images, as well as the ground-truth trajectory from the motion capture system. To evaluate the error in the estimated trajectory by comparing it with the ground-truth, we adopt the absolute trajectory error (ATE) root-mean-square error metric (RMSE) as proposed in [112]. $\{P_i^{est} \in \mathbb{SE}_3\}_{i=1}^n$ and $\{P_i^{gt} \in \mathbb{SE}_3\}_{i=1}^n$ are sequences of camera poses from the estimated trajectory and from the ground truth trajectory, respectively. The error E_i between P_i and Q_i at time step i is then given by:

$$E_i = (P_i^{gt})^{-1} S P_i^{est} \tag{3.35}$$

where S the rigid-body transformation found by the Horn method [51] to align the trajectories. The ATE is defined as the root mean square error from error matrices:

$$\mathrm{ATE}_{rms} = \left(\frac{1}{n} \sum_{i=1}^{n} \| \mathrm{trans}(E_i) \|^2 \right)^{1/2} \tag{3.36}$$

where $\mathrm{trans}(E_i)$ are the translational components from the relative pose error E_i.

Table 3.1: Comparison of the absolute trajectory error ATE_{rms} [m] on the TUM RGB-D dataset [112]. freiburg3_large_cabinet (freiburg3_lc); $E^{pa}_{icp\_rgb\_sem}$ (Ours)

	E_{icp}	E_{rgb}	E_{icp_rgb}	$E_{icp\_rgb\_sem}$	Ours
freiburg1_desk	0.030	0.031	0.020	0.017	**0.016**
freiburg1_room	0.074	0.075	0.068	0.064	**0.060**
freiburg1_teddy	0.086	0.087	0.083	0.078	**0.071**
freiburg1_desk2	0.025	0.024	0.022	0.018	**0.017**
freiburg2_desk	0.082	0.088	0.071	0.065	**0.060**
freiburg2_xyz	0.013	0.014	0.011	0.009	**0.009**
freiburg3_lc	0.112	0.105	0.099	0.043	**0.040**
MEAN	0.060	0.061	0.053	0.042	**0.039**

Table 3.2: Comparison of absolute trajectory error ATE_{rms} [m] on the warehouse dataset.

	E_{icp}	E_{rgb}	E_{icp_rgb}	$E_{icp\_rgb\_sem}$	$E^{pa}_{icp\_rgb\_sem}$
warehouse_01	0.035	0.031	0.026	0.020	**0.019**
warehouse_02	0.035	0.045	0.031	0.027	**0.023**
warehouse_03	0.042	0.041	0.036	0.028	**0.025**
warehouse_04	0.033	0.031	0.022	0.016	**0.015**
warehouse_05	0.050	0.058	0.045	0.031	**0.028**
warehouse_06	0.035	0.037	0.028	0.024	**0.022**
warehouse_07	0.040	0.037	0.034	0.028	**0.026**
warehouse_08	0.050	0.053	0.041	0.034	**0.030**
warehouse_09	0.043	0.044	0.036	0.026	**0.021**
warehouse_10	0.043	0.045	0.037	0.031	**0.029**
MEAN	0.041	0.042	0.034	0.027	**0.024**

Tables 3.1 and 3.2 show the results. The best quantities are marked in bold. The evaluation shows that our proposed objective function $E_{icp\_rgb\_sem}$ out-

performs the others in all the tested sequences. This result is explainable since the strengths and the weaknesses of the error functions are different and a combination would help them to perform better. We also can see that the combined energy function E_{icp_rgb} is more suitable for achieving precise alignment than the geometric energy E_{icp} and photometric ICP E_{rgb}.

In addition, we performed an ablation study and computed the trajectory errors for our approach with fixed per-image adaptive weights, per-pixel adaptive weights, fixed weights, full and simplified semantic error functions. As we can see in Fig. 3.8 and 3.9, the full version of our approach using per-pixel adaptive weights consistently results in the lowest observed trajectory errors across all datasets. While the use of the semantic energy E_{sem_full} in Eq. 3.27 results in slightly lower ATE_{rms} errors, the computational time required (200ms) for registration is much higher than using the simplified function E_{sem} (30ms). Therefore, we decided to use the simplification in our system. A visualization of trajectories by running ElasticFusion using the combined geometric and photometric error function E_{icp_rgb} and our proposed objective function $E_{icp\_rgb\_sem}$ on two sequences freiburg1_desk2 and freiburg1_teddy in TUM RGB-D dataset [112] is shown in Fig. 3.10. Similarly, Fig. 3.11 visualizes estimated trajectories of two videos in the warehouse dataset. Note that the TUM RGB-D dataset does not contain ground-truth data for semantic segmentation. Therefore, we used data from SceneNN [56] for training the Fast-RGBD-SSWP network and then run inference for TUM images.

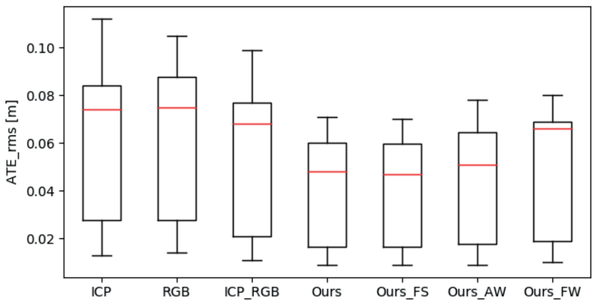


Figure 3.8: Boxplot of the ATE_{rms} in meters on the TUM RGB-D dataset (over all sequences considered in Table 3.1). We ran a number of tests on sequences of the dataset to analyze the different energy functions including (i) E_{icp} (ICP), (ii) E_{rgb} (RGB), (iii) E_{icp_rgb} (ICP_RGB), (iv) $E^{pa}_{icp\_rgb\_sem}$ with per-pixel adaptive weights and simplified function E_{sem} (Ours), (v) $E^{pa}_{icp\_rgb\_sem}$ with per-pixel adaptive weights and E_{sem_full} (Ours_FS), (vi) $E_{icp\_rgb\_sem}$ with per-image adaptive weights (Ours_AW), (vii) $E_{icp\_rgb\_sem}$ with fixed weights (Ours_FW). In each box the red central line is the median, the box edges the 25th and 75th percentiles and the whiskers extend to the minimum and maximum estimates.

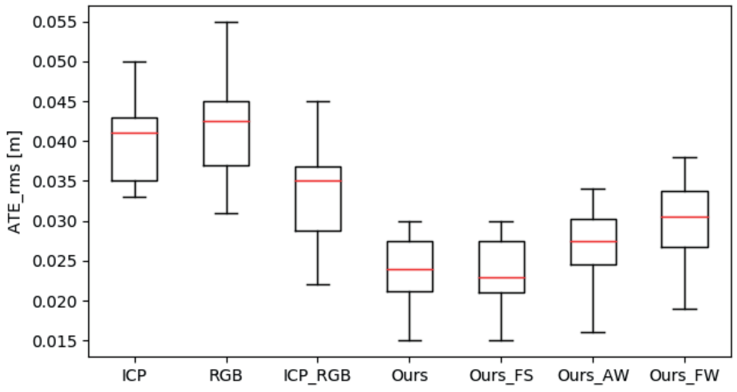


Figure 3.9: Boxplot of the ATE_{rms} in meters on the warehouse dataset. We ran a number of tests on sequences of the dataset to analyze the different energy functions including (i) E_{icp} (ICP), (ii) E_{rgb} (RGB), (iii) E_{icp_rgb} (ICP_RGB), (iv) $E^{pa}_{icp\_rgb\_sem}$ with per-pixel adaptive weights and simplified function E_{sem} (Ours), (v) $E^{pa}_{icp\_rgb\_sem}$ with per-pixel adaptive weights and E_{sem_full} (Ours_FS), (vi) $E_{icp\_rgb\_sem}$ with per-image adaptive weights (Ours_AW), (vii) $E_{icp\_rgb\_sem}$ with fixed weights (Ours_FW). In each box the red central line is the median, the box edges the 25th and 75th percentiles and the whiskers extend to the minimum and maximum estimates.

3.6.3 Reconstruction Results

In the context of robot manipulation, a reconstruction system is required to build a persistent and accurate 3D map of reconstructed objects. The quality of the model is vital for picking, and in particular small detailed parts of the objects of interest are more important than having an even precision throughout the environment. Hence, we can not evaluate the system by looking at trajectory estimates alone. In this section, we address the surface reconstruction quality aspect in more detail.

In order to evaluate surface reconstruction quality, we compare the reconstructed point cloud of each object to its ground truth 3D model. For every object present in the scene, we need to align the reconstructed object point cloud O to the ground truth model G. To achieve this, we utilize a human-assisted alignment tool [78]. More specifically, We first manually identify each object in the scene and then select the corresponding mesh object model. Then we provide an initial alignment by clicking three points on the object in the reconstructed point cloud, and then clicking roughly the same three points in the object model as shown in Fig. 3.12. Given a rough alignment from the previous step, a cropped point cloud is taken from the points within 1cm of the

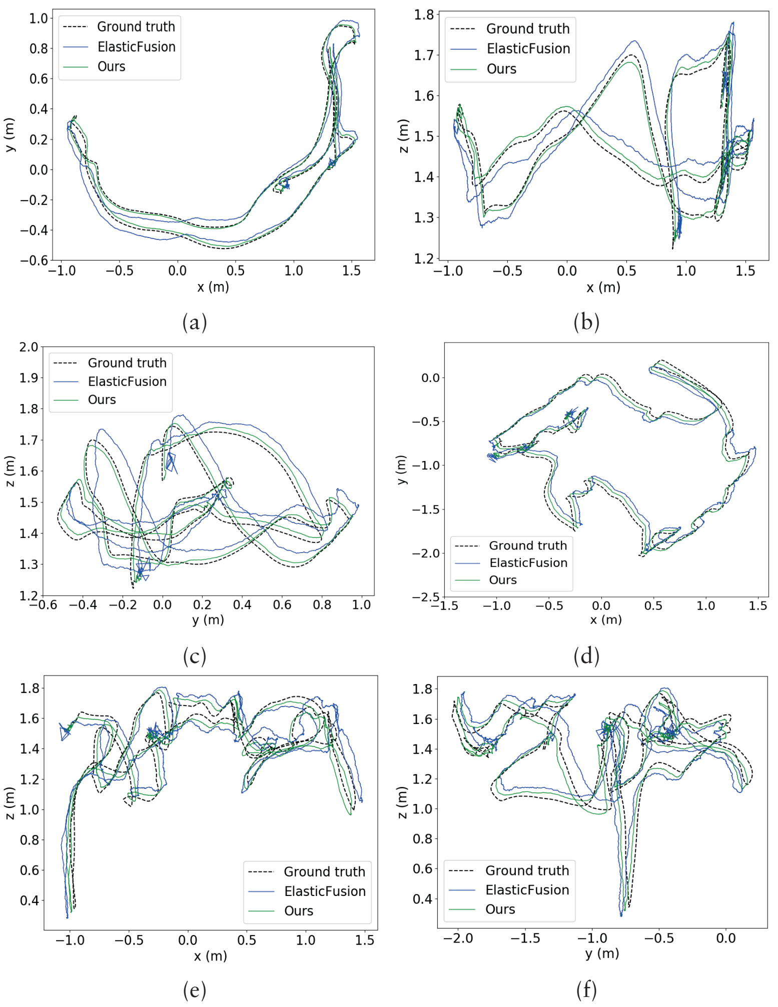
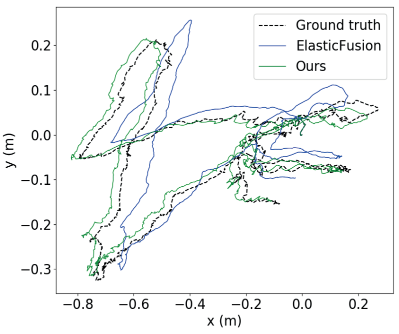
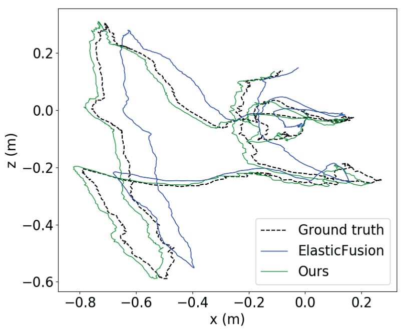


Figure 3.10: The result trajectories estimated by ElasticFusion (using E_{icp_rgb}) and our proposed approach (using $E_{icp\_rgb\_sem}$) compared to the ground truth of two sequences in the TUM RGB-D dataset [112]. Ground truth and camera trajectories projected to 2D: (a-c) `freiburg1_desk2` sequence, (d-f) `freiburg1_teddy` sequence.

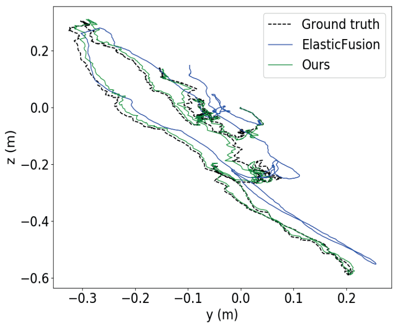
roughly aligned model. Finally, in order to obtain a fine alignment, ICP method is employed to align this cropped point cloud to the model.



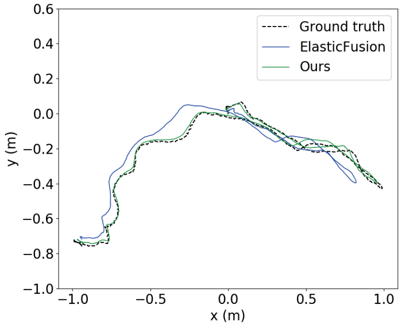
(a)



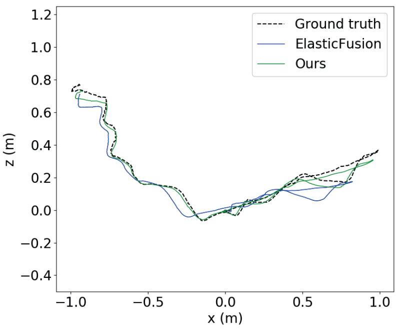
(b)



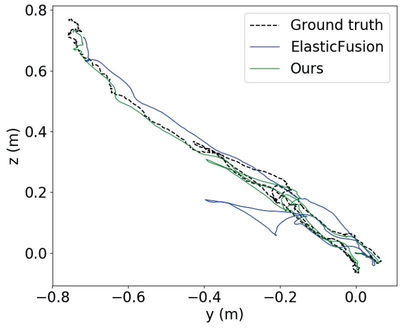
(c)



(d)

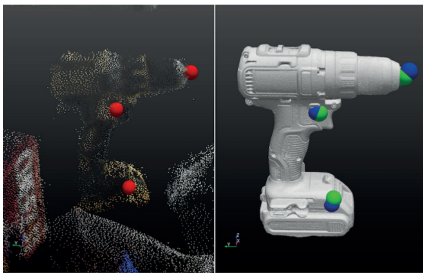


(e)

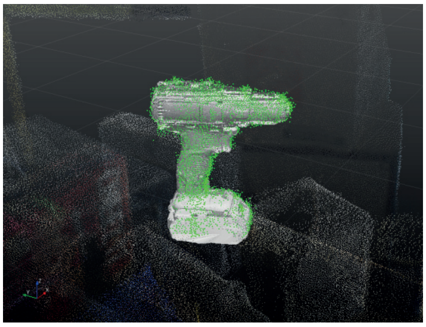


(f)

Figure 3.11: The result trajectories estimated by ElasticFusion (using E_{icp_rgb}) and our proposed approach (using $E_{icp\_rgb\_sem}$) compared to the ground truth of two videos in the warehouse dataset. Ground truth and camera trajectories projected to 2D: (a-c) video 1, (d-f) video 2.



(a)

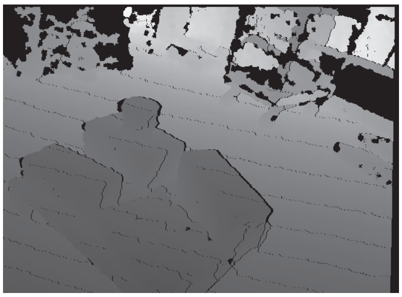


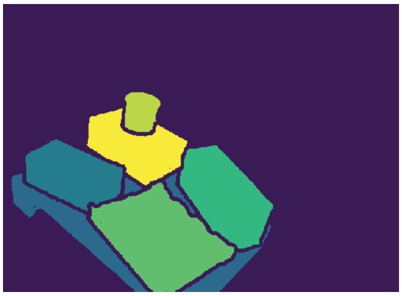
(b)

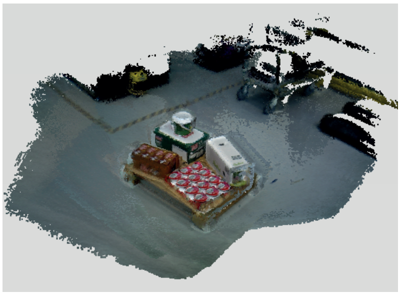
Figure 3.12: Human Assisted alignment tool (model-to-reconstruction) [78]. (a) Initial alignment using 3 clicks; (b) Alignment result.

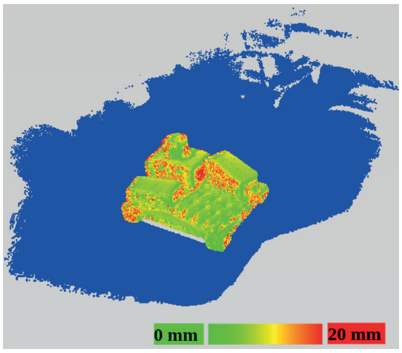
At this stage, we can start evaluating the quality of reconstructed point clouds. We project every vertex from O onto G and compute the distance between the original vertex and its projection. Finally, we calculate and report the mean distance μ_d over all model points. The results of this evaluation on the reconstruction datasets are summarised in Table 3.3. Qualitative results are shown in Fig. 3.13. Our system using $E^{pa}_{icp\_rgb\_sem}$ consistently results in the lowest reconstruction errors over all objects. From this comparison, it is evident that the proposed approach benefits greatly from the use of the proposed joint cost function. We observe an increase in accuracy is achieved when more segmented objects appeared in the reconstructed environment, suggesting that our system makes efficient use of the available semantic information to improve surface reconstruction quality. In other words, when the number of objects of interest increases the semantic probability map becomes more textured, which leads to a better reconstruction performance.

(a) Scene

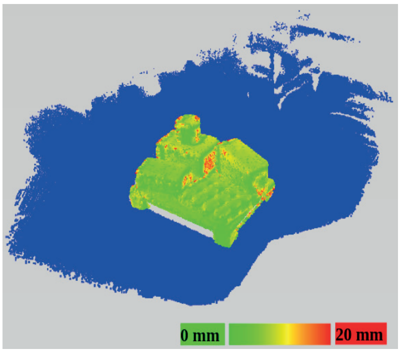

(b) Depth


(c) Segmentation


(d) 3D map



(e) ElasticFusion



(f) Ours

Figure 3.13: Heat maps showing reconstruction error of ElasticFusion (EF) and our proposed method. Color-coded visualization of point-wise distance error ranging from 0 mm to 20 mm. Points belonging to the background are color-coded with blue only for visualization purpose.

Table 3.3: Comparison of surface reconstruction error (mm) results on the warehouse objects. We ran a number of ablations to analyze the proposed objective function including (i) $E_{icp\_rgb\_sem}$ with per-image adaptive weights and E_{sem_full} (Ours*), (ii) $E_{icp\_rgb\_sem}$ with per-image adaptive weights and the simplified function E_{sem} (Ours^{-}), and (iii) $E^{pa}_{icp\_rgb\_sem}$ with per-pixel adaptive weights and the simplified function E_{sem} (Ours).

	E_{icp}	E_{rgb}	E_{icp_rgb}	Ours*	Ours^{-}	Ours
001_frasvaf_box	9.5	9.2	8.3	5.5	5.6	5.0
002_small_jacky_box	9.1	9.8	6.5	5.8	6.0	5.3
003_jacky_box	8.1	8.3	6.6	5.1	5.3	4.7
004_skansk_can	9.6	9.9	7.9	6.2	6.3	5.6
005_sotstark_can	9.8	9.0	7.3	5.2	5.2	4.5
006_onos_can	9.6	9.4	8.1	6.0	6.0	5.4
007_risi_frutti_box	6.5	7.1	4.1	4.2	4.2	4.0
008_pauluns_box	8.8	8.8	5.8	4.9	5.0	4.5
009_tomatpure	9.2	9.5	7.4	5.1	5.2	4.7
010_pallet	13.1	12.5	11.7	7.6	7.7	7.4
011_half_pallet	14.2	13.8	12.5	7.6	7.8	7.4
MEAN	9.8	9.7	8.0	5.7	5.8	5.3

3.7 Discussion

In this chapter, we have presented and evaluated algorithms for registration of RGB-D images. Our main contribution is to show that by combining geometric, appearance, and semantic cues in an adaptively weighted sum we are able to obtain reliable camera tracking and state-of-the-art surface reconstruction. We also introduced a newly collected warehouse object dataset for the evaluation of RGB-D reconstruction and object pose estimation systems. This dataset is more challenging than other publicly available datasets due to recorded environments posing more complex problems, including low illumination inside shelves, low-texture and symmetric objects, clutter, and occlusions. We have provided an extensive evaluation on a common benchmark TUM RGB-D dataset and our warehouse dataset. Experimental results confirmed that the proposed system achieves improvements over state-of-the-art methods in terms of camera pose estimation and surface reconstruction. The results confirm that the developed system is able to produce a high-quality dense map with robust tracking. We believe that the accurate scene reconstruction will open the way to new applications regarding autonomous robotic manipulation. Indeed, in the next chapter, we exploit this high quality reconstruction system to boost the performance of 6D object pose estimation.

Chapter 4
Object Pose Estimation with Semantic Mapping

4.1 Introduction

The problem of recognizing objects and estimating their 6D poses has drawn interest from the research community over the last two decades because of its practical value. Especially in robotic manipulation, localizing objects in 3D from images is an important task towards highly complex autonomous systems. The ability to estimate the 3D location and 3D orientation of objects, provides useful information for reasoning about contact, physics, and occlusion among objects. Many approaches were introduced in the past. However, accurate 6D object pose estimation still remains a largely unsolved problem, especially when objects are occluded.

Conventional methods are mainly based on the matching of hand-crafted local or global features to extract the correspondence between images (2D images or 3D point clouds) and object mesh models. Global feature-based approaches utilize the whole geometric appearance of the object surface to define a single feature vector that effectively and concisely describes the entire 3D object [105, 131, 2]. On the contrary, the local feature-based methods exploit the geometric properties around specific keypoints [40, 26, 1]. While global methods are able to handle objects with self-similar surface parts, such as planar patches, spheres, and cylinders, local approaches are more suitable for detecting and estimating the pose of complex objects in cluttered scenes. Nonetheless, due to the time-consuming multi-stage processing for feature extraction, generating coarse pose hypotheses, and refining the coarse poses, it is difficult for hand-crafted feature-based methods to satisfy the requirements of accurate pose estimation and fast inference simultaneously. More recently, with the explosive growth of advances in machine learning, especially deep learning, Deep Neural Network (DNN) based approaches have been introduced into this task [133, 115, 121]. Compared to conventional hand-crafted feature extractors,

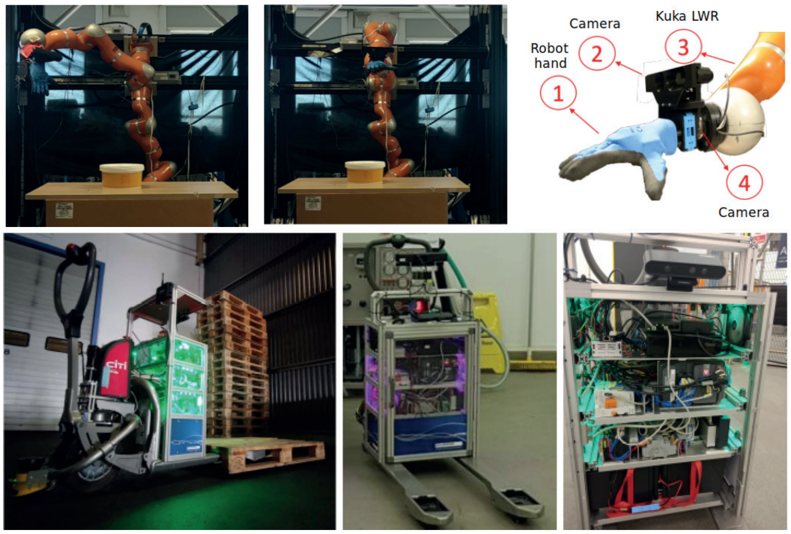


Figure 4.1: Robots with an attached camera. (top row) an eye-in-hand configuration, an ASUS RGB-D camera is mounted on top of a robot hand. In this way, the robot can position the camera in many different locations. (bottom row) a mobile robot equipped with an ASUS RGB-D camera allows a more flexible and effective solution.

DNNs capture different scale information in different layers, and extract robust and discriminative features. These techniques demonstrate a significant improvement of inference time and the accuracy of 6D object pose estimation.

Although DNN-based methods have shown promising results, due to the limitation of single-view-based pose estimation they still suffer significantly from the ambiguity of object appearance and occlusions in cluttered scenes, which are very common in practice. These methods consider objects independently and estimate their poses using a single input (RGB or RGB-D) image. However, scenes are often composed of different object instances and multiple images of the scene can easily be captured by a single moving camera, or in a multiple-camera setup. For instance, in a robot manipulation setting, a camera can be attached on the robot end effector, in which the camera is moved continuously and posed with different gestures by the robot within its working area as in Fig. 4.1. From this observation, some works [110, 66, 72] have addressed the problem of the ambiguity of object appearance and occlusion by combining information from multiple views. The common approach used in multi-view methods is employing a single-view algorithm on each of the images and combining the resulting output. However, the challenge here is that estimated object poses from individual images cannot easily be expressed in the same global coordinate system because the camera poses are unknown. Another problem is that since most of the object pose estimation approaches rely on segmentation results for pose prediction, their accuracy highly depends on the performance of the segmentation framework used. As in DNN-based approaches, if the input to a segmentation network contains an occluder, the occlusion significantly influences the network output. Similarly, estimating object poses directly from raw RGB-D data often gives poor results because of noise from the sensor. Hence, recovering 6D object pose in cluttered scenes is still an open problem.

In this chapter, we present a method for multi-view 6D object pose estimation using accurate semantic reconstruction with an RGB-D camera. We overcome the challenge of finding unknown camera poses by utilizing the registration algorithm based on geometric, photometric and semantic cues presented in the previous chapter. To address the issue regarding measurement noise, our approach explores performing 6D object pose estimation from multiple viewpoints supported by a high-quality semantic reconstruction system. Specifically, we integrate deep-learning-based semantic segmentation, instance segmentation, and 6D object pose estimation into a state-of-the-art RGB-D mapping system [129]. To mitigate the adverse effects of measurement noise, instead of directly using raw depth and color frames captured by the camera for object pose estimation, we employ the surfel-splatted predicted depth map and the color image of our accurate reconstruction scene model. Finally yet importantly, rather than directly operating on masks from the segmentation network, we use predicted 2D masks that are obtained by reprojecting the current reconstructed scene model. Thereby, our object pose estimation method benefits from the use of more accurate segmentation results.

In the next section, we overview some of the recent developments in semantic mapping. Section 4.3 describes the proposed multi-view object pose estimation with semantic mapping, which is a central concept in this chapter. The results are then presented and analyzed in Section 4.4, followed by a summary of the major contributions.

4.2 Semantic Reconstruction

Fusing semantic along with geometric information within a 3D reconstructed map is a promising approach to enable robots to better understand a 3D scene. The inclusion of rich semantic information and 6D poses of object instances within a dense map is useful for robots to effectively operate and interact with objects. In the case of robotic manipulation, providing accurate object poses together with semantic information is crucial for robots that have to manipulate the objects around them in diverse ways. To accurately grasp selected objects and avoid collisions with neighboring obstacles in the workspace, the reconstruction process needs to produce a high-quality map of the working environment. The addition of semantic information enables a much greater range of functionality than geometry alone.

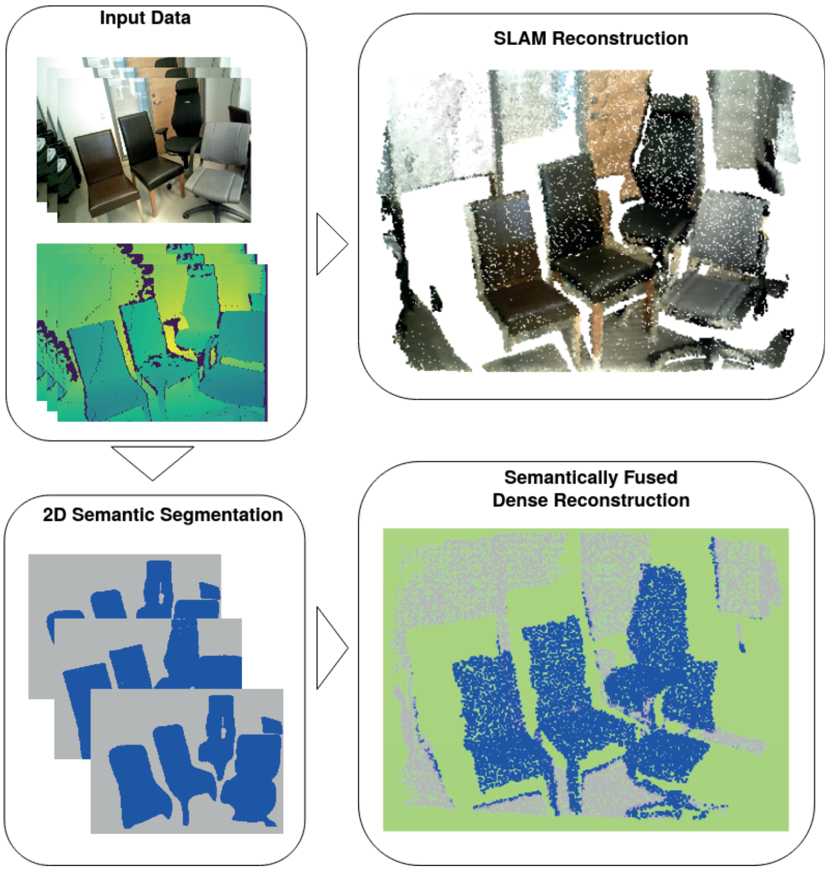


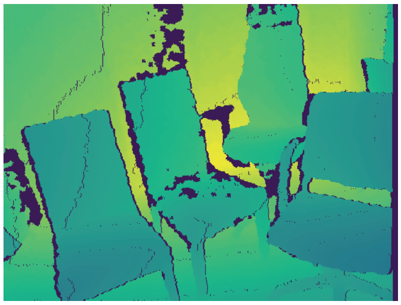
Figure 4.2: Semantic reconstruction pipeline in [80]. A sequence of color images and depth maps are used as input images to build a 3D scene model, and to generate 2D semantic labels as well as a set of probability prediction maps. These maps are fused into the final dense semantic model via Bayesian updates.

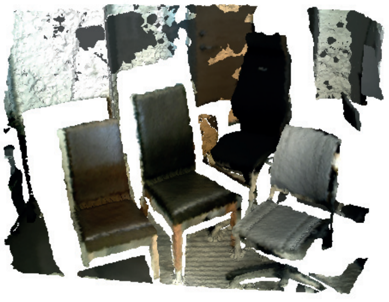
Recent advances in deep learning-based semantic segmentation have enabled the integration of rich semantic information within real-time Simultaneous Localization and Mapping (SLAM) systems. A number of semantic mapping systems have been developed [80, 47, 132]. Hermans et al. [47] utilize Random Decision Forests to achieve semantic pixel-wise image labeling and fuse them in a classic Bayesian framework. Previous work by McCormac et al. [80] aimed at combining Convolutional Neural Networks and ElasticFusion [129] to obtain semantic-aware 3D reconstruction as shown in Fig. 4.2. The correspondences between frames are estimated by the SLAM system. Meanwhile, their CNN architecture adopts a Deconvolutional Semantic Segmentation network [89] to generate a pixel-wise semantic map for incoming images. Unlike the original architecture [89], this system incorporates depth information to obtain a higher accuracy than the pretrained RGB network. The authors reported that fusing multiple predictions led to a significant improvement in semantic labeling and it is the first real-time capable approach suitable for interactive indoor scene scanning and labeling. Likewise, SegICP-DSR [132] fuses RGB-D observations into a semantically-labeled point cloud for object pose estimation using adversarial networks and ElasticFusion. There is, however, one significant difference. SegICP-DSR employs the semantic label difference instead of a photometric error when formulating the alignment objective function. Then, a semantically-labeled point cloud can be directly obtained from the reconstruction process without an extra update step. The addition of semantic information enables a much greater range of functionality than geometry alone. However, since the above systems only consider class labels, they are limited to scenarios with single object instances per scene and may degenerate performance in case multiple objects of the same type are present.

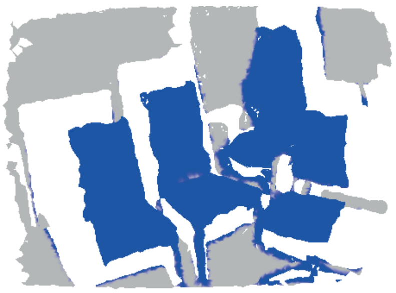
A number of other works have been proposed to overcome the limitation of semantic reconstruction [79, 86, 103]. These approaches generate the reconstructed model of an environment enriched with semantic information in the form of object instances, called object-oriented semantic reconstruction or instance-aware semantic reconstruction. Fig. 4.3 illustrates differences between conventional geometric reconstruction, semantic reconstruction and object-oriented reconstruction. The work of McCormac et al. Fusion++ [79] aimed to produce multiple semantically labeled maps of object instances without a dense representation of the entire static scene. Fusion++ uses Mask R-CNN instance segmentation to initialize dense per-object TSDF reconstructions with object size-dependent resolutions. For camera tracking, Fusion++ takes an approach similar to KinectFusion using projective data association and a point-to-plane error. Note that apart from object level maps, Fusion++ also maintains a coarse background TSDF to assist frame-to-model tracking. While the authors evaluated the trajectory error of the developed system against the baseline approach of simple coarse TSDF odometry, the reports did not provide a comparison with other photometry or semantics-aware state-of-the-art approaches.

Similarly, MaskFusion [103] is a real-time, object-aware, semantic and dynamic RGB-D SLAM system. It combines geometric segmentation running on every frame and instance segmentation using Mask R-CNN computed for select keyframes. The geometric segmentation algorithm acquires object boundaries based on an analysis of depth discontinuities and surface normals, while Mask R-CNN is used to provide object masks with semantic labels. Camera poses are estimated by minimizing a joint geometric and photometric error function as presented in [129]. The reported results demonstrate that while MaskFu-

(a) RGB image


(b) Depth image


(c) 3D reconstruction


(d) Semantic reconstruction

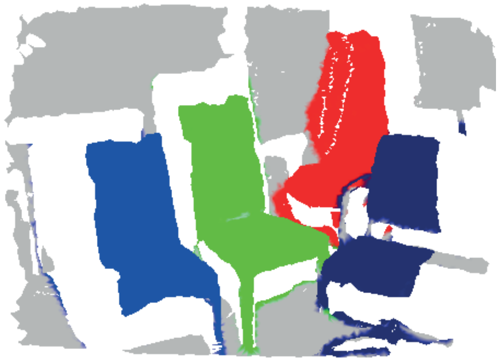

(e) Object-oriented semantic reconstruction

Figure 4.3: An example indicates the difference between semantic reconstruction and object-oriented semantic reconstruction [86]. The semantic map in (c) can distinguish different object classes but not object instances. The monitors are represented as the same color and we can not discriminate between them. On the contrary, in (d) object-oriented semantic reconstruction or instance-aware semantic map is able to differentiate individual objects.

sion outperforms a set of baseline state-of-the-art algorithms in highly dynamic scenes, ElasticFusion performs best on static and moderately dynamic scenes.

While previous approaches simply fuse semantic information into global scene models as an additional attribute, our work discovers the use of semantic cues to boost the performance of reconstruction and object pose estimation. In Chapter 3, we have shown that the use of semantic cues can improve the accuracy of camera pose estimation and scene reconstruction. In the next section, we explore the performance of 6D object pose estimation from multiple viewpoints supported by a high-quality semantic reconstruction system that is built upon the system in Chapter 3.

4.3 Object Pose Estimation with An Accurate Semantic Map

Our proposed pipeline is visualized in Fig. 4.4. The input RGB-D data is processed through a semantic segmentation module (as presented in chapter 3), followed by camera pose tracking, and finally a data fusion stage. In a separate thread, RGB keyframes are processed by an instance segmentation framework (Mask R-CNN [44]) and the detections are filtered and matched to the existing instances in the 3D map. When no match occurs, new object instances are created. Note that our pipeline does not specifically limit the choice of instance segmentation frameworks. Mask-RCNN can be replaced by a different instance segmentation approach of comparable quality. The final component is a 6D object pose estimator that exploits multiple views of the same instance and our high-quality reconstruction to accurately predict the pose of objects.

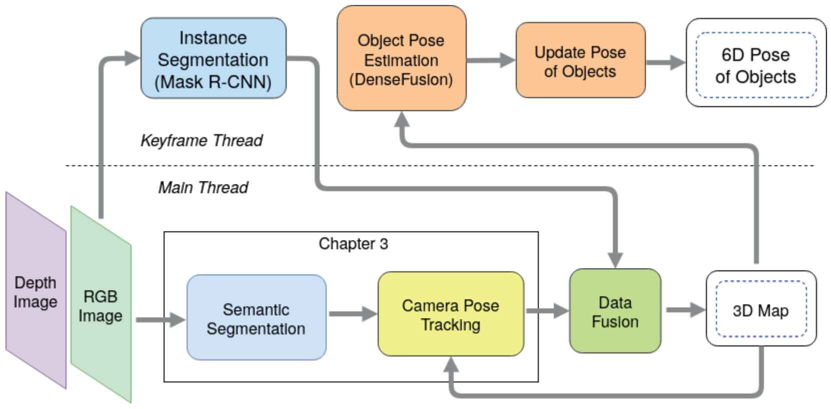
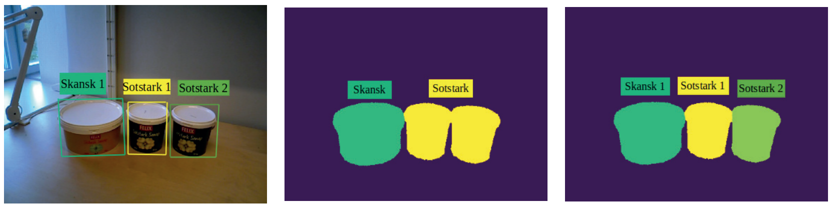


Figure 4.4: Overview of the proposed system.

4.3.1 Review of Mask R-CNN

Different from object detection and semantic segmentation, the goal of instance segmentation is to classify each pixel of an image into a fixed set of categories and differentiate object instances (see Fig. 4.5). Mask R-CNN [44] is a CNN-based framework for instance segmentation. It takes an image as input and outputs a class label, a bounding box offset, and a mask for each candidate object. Fig 4.6 shows the architecture of Mask R-CNN. Its procedure consists of two stages. In the first stage, candidate object bounding boxes are proposed by a Region Proposal Network (RPN). In the second stage, classification, bounding-box regression, and mask prediction are performed in parallel on each small feature map. To speed up inference and improve accuracy, the mask branch is applied to the highest scoring 100 detection boxes after running the box prediction. The mask branch predicts a binary mask from each RoI using an FCN architecture [74]. The binary mask is a single $m \times m$ output regardless of class, which is generated by binarizing the floating-number mask or soft mask at a threshold of 0.5. The output of Mask-RCNN including class probabilities and masks is then used in the object pose estimation stage.



(a) Object detection (b) Semantic segmentation (c) Instance segmentation

Figure 4.5: An illustration of differences between object detection, semantic segmentation, and instance segmentation.

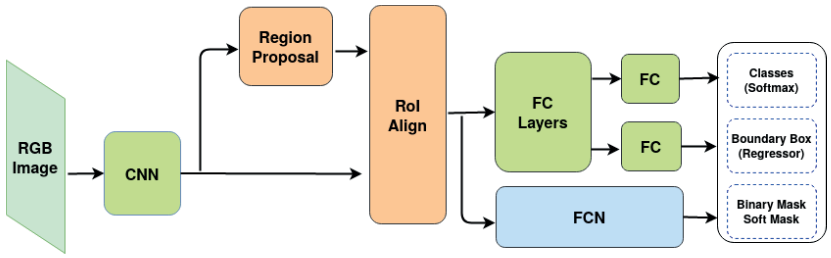


Figure 4.6: Mask R-CNN architecture for instance segmentation. CNN, RoI, FCN, and FC stand for convolutional neural network, region of interests, fully convolutional networks, fully connected respectively.

4.3.2 Incremental Semantic Map Fusion

To perform camera tracking, our mapping system maintains a fused surfel-based model of the environment (similar to the model used by ElasticFusion [129]). Here we borrow and extend the notation proposed in the original ElasticFusion paper. The model is represented by a cloud of surfels $\mathcal{M}^s$, where each surfel consists of a position $p \in \mathbb{R}^3$, normal $n \in \mathbb{R}^3$, color $c \in \mathbb{N}^3$, initialization timestamp t_0 and last updated timestamp t. In addition we maps each element of the 3D map (surfel) to a pair $(l_s, \mathbf{o}_s) \in \mathcal{L} \times \mathbb{N}$, where l_s represents the semantic class of surfel s and $\mathbf{o}_s$ represents its object instance id. $\mathcal{L}$ is a predetermined set of L semantic classes encoded by $\mathcal{L} := \{0, ..., L-1\}$. We estimate an incremental transformation $\hat{\xi}$ between a newly captured RGB-D image at time t and the previous sensor pose at time $t-1$ by minimizing a joint optimization objective as presented in Chapter 3.

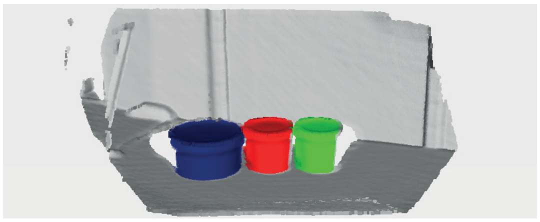
Data association: Given an RGB-D frame at time step t, each mask M from Mask R-CNN must be associated with an instance in the 3D map. Otherwise, it will be assigned as a new instance. To find the corresponding instance, we use the tracked camera pose and existing instances in the map built at time step $t-1$ to predict binary masks via splatted rendering. The overlap percentage between the mask M and a predicted mask $\hat{M}$ for object instance $\mathbf{o}$ is computed as $\mathbb{U}(M, \hat{M}) = \frac{M \cap \hat{M}}{\hat{M}}$. Then the mask M is mapped to object instance $\mathbf{o}$ which has the predicted mask $\hat{M}$ with largest overlap, where $\mathbb{U}(M, \hat{M}) > 0.3$. Fig 4.7 shows a series of 4 frames illustrating the incremental semantic map fusion of the proposed system.

To efficiently store class probabilities, we propose to assign an object instance label $\mathbf{o}$ to each surfel and then this label is associated with a discrete probability distribution over potential class labels, $P(L_{\mathbf{o}} = l_i)$ over the set of class labels, $l_i \in \mathbb{L}$. In consequence, we need only one probability vector for all surfels belonging to the same object entity. This makes a big difference when the number of surfels is much larger than the number of classes. To update the class probability distribution, recursive Bayesian update may be used as in [47]. However, this scheme often results in an overly confident class probability distribution that contains scores unsuitable for ranking in object detection [79]. In order to make the distribution more even, we update the class probability by simple averaging:

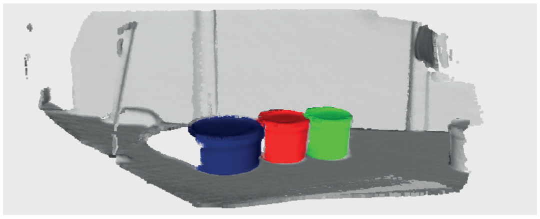
$$P(l_i | I_{1,..,t}) = \frac{1}{t} \sum_{j=1}^{t} (p_j | I_t) \tag{4.1}$$

Besides fusing main class probabilites, we enrich segmentation information on each surfel by adding the probability to account for background/object predictions from the binary mask branch of Mask R-CNN. To that end, each surfel

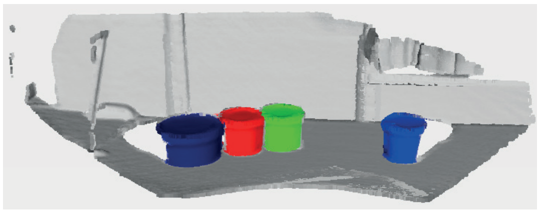
(a) Frame 66 - color image

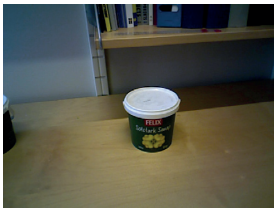

(b) 3D map at frame 66

(c) Frame 316 - color image


(d) 3D map at frame 316

(e) Frame 767 - color image


(f) 3D map at frame 767


(g) Frame 1000 - color image

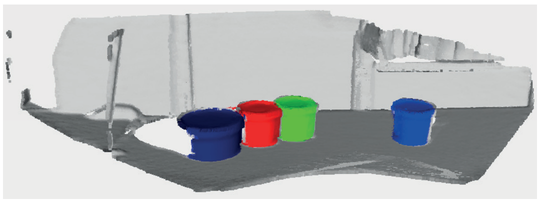

(h) 3D map at frame 1000

Figure 4.7: A series of 4 frames illustrating the incremental semantic map fusion of the proposed system.

in our 3D map has a non-background (object) probability attribute p_o. As presented in [44] the binary mask branch first generates an $m \times m$ floating-number mask which is then resized to the RoI size, and binarized at a threshold of 0.5. Therefore, we are able to extract a per-pixel non-background probability map with the same image size 480×640. Given the RGB-D frame at time step t, a non-background probability $p_o(I_t)$ is assigned to each pixel. Camera tracking and the 3D back projection introduced in Chapter 3 enables us to update all the surfels with the corresponding probability as following:

$$p_o = \frac{1}{t} \sum_{j=1}^{t} p_j(I_t) \tag{4.2}$$

Segmentation Improvement: Despite the power and flexibility of Mask R-CNN, it frequently misclassifies object boundary regions as background. In other words, the detailed structures of an object are often lost or smoothed. Thus, there is still much room for improvement in segmentation. We observe that many of the pixels in the misclassified regions have non-background probability just slightly smaller than 0.5, while the soft probabilities mask for real background pixel is often far below the threshold. Based on this observation, we expect to achieve a more accurate object-aware semantic scene reconstruction by considering the non-background probability of surfels within a frame sequence. With this goal, each possible surfel s ($0.4 < p_o < 0.5$) is associated with a confidence $\vartheta(s)$. If a surfel is identified for the first time, its associated confidence is initialized to zero. Then, when a new frame arrives, we increment the confidence $\vartheta(s) \leftarrow \vartheta(s) + 1$ only if the corresponding pixel of that surfel satisfies 2 criteria: (i) its non-background probability is greater than 0.4; (ii) there is at least one object pixel inside its 8-neighborhood. After n frames, if the confidence $\vartheta(s)$ exceeds the threshold σ_{object}, we assign surfel s to the closest instance. Otherwise, $\vartheta(s)$ is reset to zero.

4.3.3 Multi-view Object Pose Estimation

Given an RGB-D frame sequence, the task of 6D object pose estimation is to estimate the rigid transformation from the object coordinate system $\mathcal{O}$ to a global coordinate system $\mathcal{G}$. We assume that the 3D model of the object is available and the object coordinate system is defined in the 3D space of the model. The rigid transformation consists of a 3D rotation $R(\omega, \varphi, \psi)$ and a 3D translation $T(X, Y, Z)$. The translation T is the coordinate of the origin of $\mathcal{O}$ in the global coordinate frame $\mathcal{G}$, and R specifies the rotation angles around the X-axis, Y-axis, and Z-axis of the object coordinate system $\mathcal{O}$.

Contrary to the problem setting considered by classical single-view-based approaches, robots usually observe the same instances of objects in their environment several times and from disparate viewpoints. Thus, we explore performing object pose estimation from multiple viewpoints, under the conjecture that combining multiple predictions can improve the robustness of an object pose estimation system. For every single frame, we utilize a modified DenseFusion [121] object pose estimation module to predict the position and orientation of objects in 3D space. A key distinction between our approach and DenseFusion is that instead of directly operating on masks from segmentation, we use predicted 2D masks, depth and RGB images that are obtained by reprojecting of the current surfel map $\mathcal{M}^s$. Our semantic mapping system leads to an improvement in the 2D instance labeling over the baseline single frame predictions generated by Mask R-CNN. As a result, our object pose estimation method benefits from the use of more accurate segmentation results as well as a high-quality scene model. The predicted poses are then transferred to the global coordinate system and serve as measurement inputs for an extended Kalman filter (EKF) to estimate an optimal pose of each object.

Object pose update: For each frame at time t, the estimates obtained by DenseFusion and camera motions from the registration stage are used to compute the pose of each object instance with respect to the global coordinate system $\mathcal{G}$. The pose is then used as a measurement update in a Kalman filter to estimate an optimal 6D pose of the object. Since we assume that the measured scene is static over the reconstruction period, the object's motion model is constant. The state vector of the EKF combines the estimates of translation and rotation:

$$\mathbf{x} = [X \quad Y \quad Z \quad \phi \quad \varphi \quad \psi]^\top \tag{4.3}$$

Let x_t be the state at time t, $\hat{\mathbf{x}}_t^-$ denote the predicted state estimate and P_t^- denote predicted state covariance at time t given the knowledge of the process and measurement at the end of step $t-1$, and let $\hat{\mathbf{x}}_t$ be the updated state estimate at time t given the pose estimated by DenseFusion z_t. The EKF consists of two stages: prediction and measurement update (correction) as follows.

Prediction:

$$\hat{\mathbf{x}}_t^- = \hat{\mathbf{x}}_{t-1} \tag{4.4}$$
$$P_t^- = P_{t-1} \tag{4.5}$$

Measurement update:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- \oplus K_t(z_t \ominus \hat{\mathbf{x}}_t^-) \tag{4.6}$$
$$K_t = P_t^-(P_t^m + P_t^-)^{-1} \tag{4.7}$$
$$P_t = (I_{6\times 6} - K_t)P_t^- \tag{4.8}$$

Here, $\ominus$ and $\oplus$ are the pose composition operators. K_t is the Kalman gain update. The 6×6 matrix P_t^m is measurement noise covariance, computed as:

$$P_t^m = \mu I_{6\times 6} \tag{4.9}$$

where μ is the mean distance from measured object points to its 3D model transformed according to the estimated pose. The measured object points are computed from depth and mask back-projected from the current 3D map.

4.4 Evaluation

We have evaluated our system by performing experiments on the YCB-Video dataset [133] and the warehouse dataset in chapter 3. These experiments are aimed at 6D object pose estimation accuracy. In addition, we also evaluate the accuracy of segmentation masks produced by our pipeline against the accuracy achieved by Mask R-CNN and state-of-the-art semantic reconstruction frameworks.

Our pipeline is implemented in ROS Kinetic using services to call different modules. The sensor pose tracking module is implemented in C++ with CUDA. The Mask R-CNN and DenseFusion codes are based on the publicly available implementations by Matterport[1] and Wang[2]. The Fast-RGBD-SSWP network for semantic segmentation is implemented using PyTorch 1.0 and the rest of the framework is in Python. In all of the presented experimental setups, results are generated from RGB-D videos with a resolution of 640x480 pixels.

4.4.1 Training Details

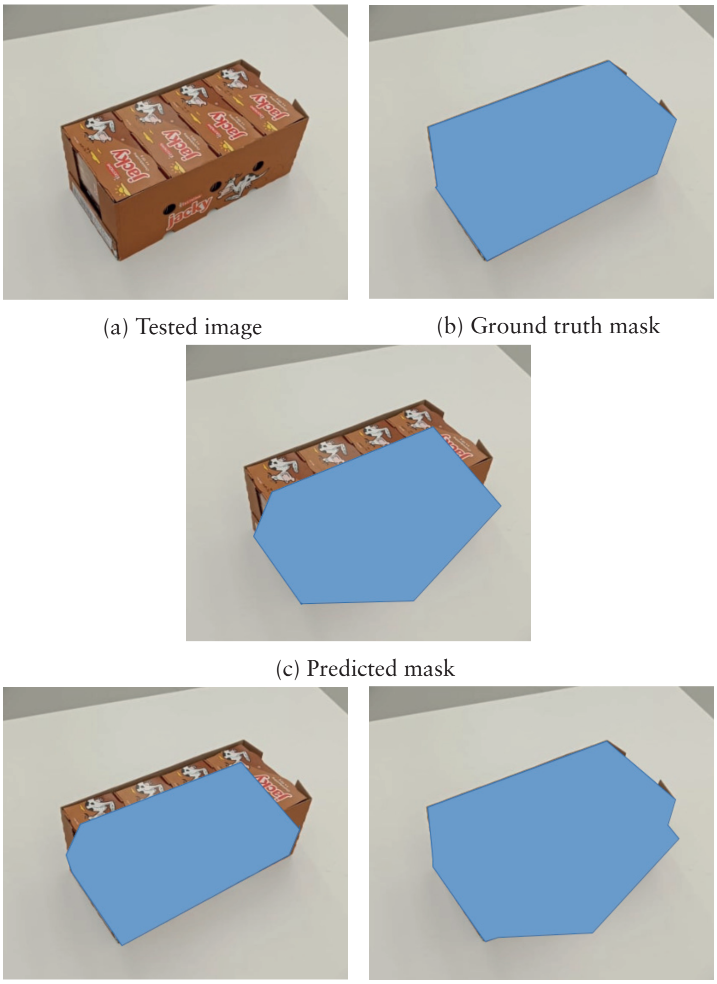
The CNNs for instance segmentation was initialized with weights pre-trained on the COCO dataset [73]. We fine tuned layers of Mask R-CNN on the warehouse dataset with 11 object classes in warehouse environments (pallet and boxes) and on a portion of the YCB video data set not used in the evaluation. We trained on 1 GPU (mini-batch size is 1 image) using stochastic gradient descent with momentum of 0.9 for 40 epochs with a learning rate of 0.001. In both warehouse dataset and YCB dataset, the semantic label sets for semantic segmentation and instance segmentation are identical, $\mathcal{L}^o = \mathcal{L}$. As regards to object pose estimation, the DenseFusion network was trained for 200 epochs with a batch size of 8. Adam [63] was used as the optimizer with learning rate set to 0.0001.

4.4.2 Segmentation

In the first experiment, we compare our method to the state-of-the-art methods in instance segmentation. We report the standard COCO metrics including AP (averaged over IoU thresholds from 50% to 95%, at a step of 5%), AP_{50} (threshold 0.5), and AP_{75} (threshold 0.75) [73]. A prediction is considered to be True Positive if IoU > threshold. The IoU metric measures the number of

[1]`https://github.com/matterport/Mask_RCNN`

[2]`https://github.com/j96w/DenseFusion`


(a) Tested image (b) Ground truth mask

(c) Predicted mask

(d) Intersection (e) Union

Figure 4.8: An illustration of intersection over union (IoU) calculation for evaluating instance segmentation.

pixels common between the ground truth and predicted masks divided by the total number of pixels present across both masks as illustrated in Fig. 4.8.

$$\mathrm{IoU} = \frac{\mathrm{Intersection}}{\mathrm{Union}} = \frac{\text{ground truth mask} \cap \text{predicted mask}}{\text{ground truth mask } \cup \text{predicted mask}} \tag{4.10}$$

To compare with Mask R-CNN, we generated 2D projections of 3D instance-aware semantic maps using the estimated camera pose trajectory. For a fair comparison, we used the current map at time step k to generate a 2D projection instead of a completed map. A numerical comparison over each label class are summarized in Table 4.1 and Table 4.2. Table 4.2 shows results on YCB-Video dataset in term of AP (averaged over IoU thresholds from 50% to 95%, at a step of 5%). Our approach improves the accuracy of Mask R-CNN by 18.5% and surpasses state-of-the-art semantic reconstruction frameworks MaskFusion [103] and Voxblox++ [38] in all the test objects by a large margin. Table 4.1 indicates results for 11 objects in warehouse dataset. We saw an improvement of 24.3% over the baseline Mask R-CNN with our system, from 62.8% to 87.1%. Similar to YCB objects, our proposed method also outperforms MaskFusion and Voxblox++ by a significant margin (14% and 11.3% respectively). Fig. 4.9 and Fig. 4.10 show boxplots of segmentation accuracy with different thresholds. As shown in the figures, it is firmly convinced that the proposed system outperforms other compared methods in both AP_{50}, AP_{75}, and AP. Qualitative results from this evaluation are shown in Fig. 4.11

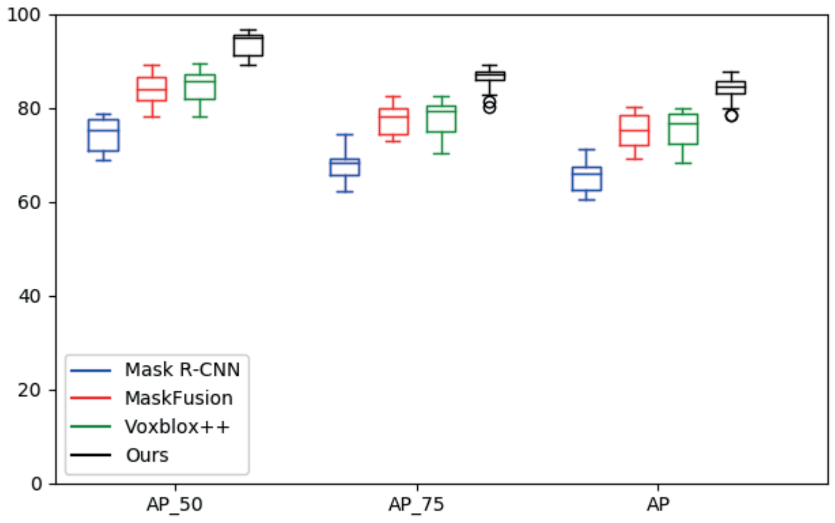


Figure 4.9: Boxplot of segmentation accuracy on YCB-Video dataset. The box edges are the 25th and 75th percentiles and the whiskers extend to the minimum and maximum values.

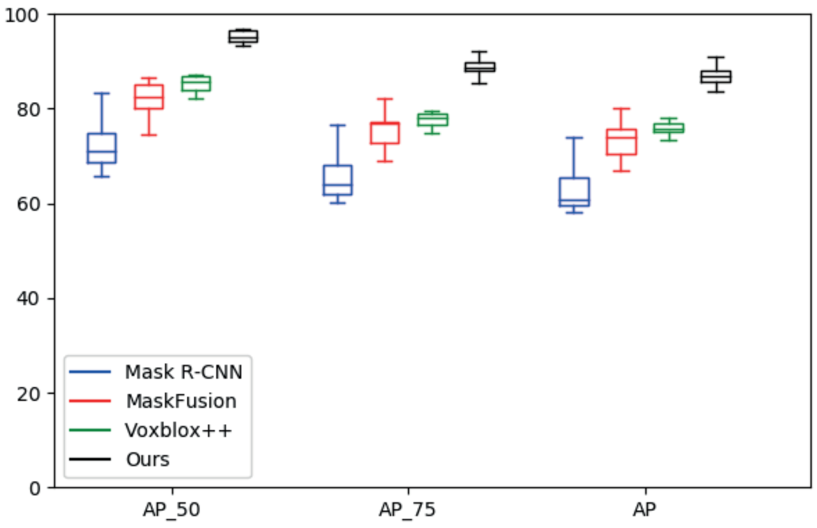


Figure 4.10: Boxplot of segmentation accuracy on the warehouse dataset. The box edges are the 25th and 75th percentiles and the whiskers extend to the minimum and maximum values.

Table 4.1: Comparison of instance segmentation accuracy (AP) results on the warehouse objects. Masks are generated by Mask R-CNN, and from 3D maps in MaskFusion [103], Voxblox++ [38], and our system.

	Mask R-CNN	MaskFusion	[38]	Ours
001_frasvaf_box	58.1	67.0	73.4	**86.3**
002_small_jacky box	74.1	75.0	76.2	**85.8**
003_jacky_box	62.1	70.0	77.1	**88.5**
004_skansk_can	58.9	74.0	73.3	**84.5**
005_sotstark_can	58.2	75.4	75.5	**86.8**
006_onos_can	60.7	76.0	77.3	**91.1**
007_risi_frutti_box	65.2	70.9	74.7	**83.7**
008_pauluns_box	66.5	76.6	75.6	**90.7**
009_tomatpure	65.8	70.0	76.6	**87.5**
010_pallet	60.1	72.3	75.8	**85.8**
011_half_pallet	60.7	80.0	78.1	**86.9**
MEAN	62.8	73.4	75.8	**87.1**

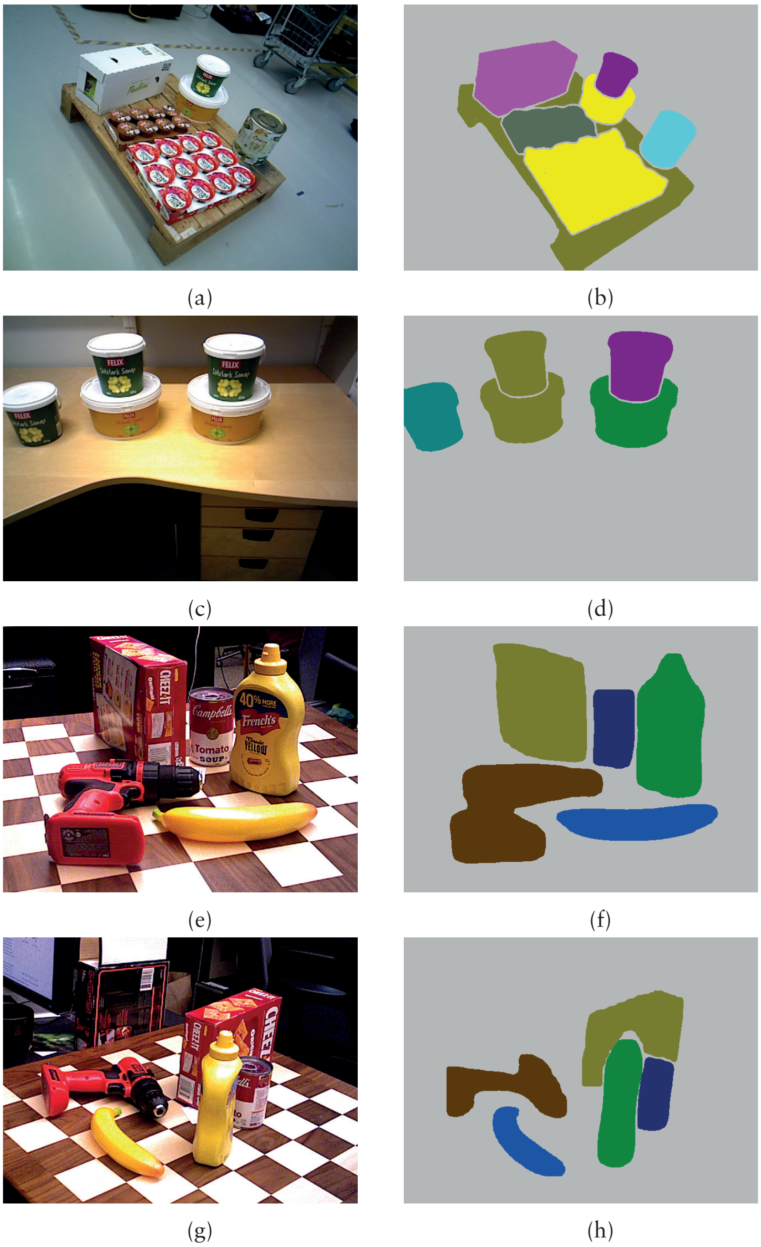


Figure 4.11: Examples of semantic segmentation result on the YCB-Video dataset and warehouse dataset.

Table 4.2: Comparison of instance segmentation accuracy (AP) results on the YCB objects. Masks are generated by Mask R-CNN, and from 3D maps in MaskFusion [103], Voxblox++ [38], and our system.

	Mask R-CNN	MaskFusion	[38]	Ours
002_master_chef_can	67.3	78.5	79.3	**85.9**
003_cracker_box	67.5	80.0	79.2	**86.7**
004_sugar_box	69.8	76.1	77.2	**87.8**
005_tomato_soup_can	66.2	75.2	75.3	**85.2**
006_mustard_bottle	67.6	76.0	79.9	**85.6**
007_tuna_fish_can	67.6	80.1	78.3	**87.8**
008_pudding_box	69.4	78.5	75.6	**84.7**
009_gelatin_box	61.4	74.4	79.1	**86.1**
010_potted_meat_can	66.1	76.6	72.4	**84.5**
011_banana	67.5	71.3	70.5	**83.2**
019_pitcher_base	60.5	75.2	77.9	**85.5**
021_bleach_cleanser	60.8	70.3	72.5	**79.8**
024_bowl	68.2	70.0	70.1	**78.5**
025_mug	66.0	78.0	79.5	**85.7**
035_power_drill	63.1	78.9	78.7	**84.4**
036_wood_block	63.6	69.3	70.5	**80.9**
037_scissors	62.6	70.0	70.2	**83.8**
040_large_marker	61.3	72.0	73.1	**81.7**
051_large_clamp	61.6	80.0	77.9	**85.4**
052_extra_large_clamp	71.3	73.0	68.2	**78.5**
061_foam_brick	65.5	74.2	76.8	**83.6**
MEAN	65.5	75.1	75.3	**84.0**

4.4.3 Pose Estimation Results

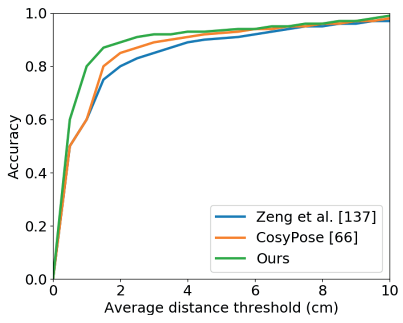
The error of an estimated pose $\hat{P}$ with respect to the ground-truth pose $\bar{P}$ of an object model M is measured by the most widely used pose-error function Average Distance of Model Points (ADD) [48]. The error is calculated as the average distance from vertices of the object model in the ground-truth pose to vertices of the model in the estimated pose. Given the ground truth rotation $\bar{R}$ and translation $\bar{T}$ and the estimated rotation $\hat{R}$ and translation $\hat{T}$, the average distance is defined as:

$$\text{ADD} = \frac{1}{m} \sum_{x \in M} \| (\bar{R}x + \bar{T}) - (\hat{R}x + \hat{T}) \| \tag{4.11}$$

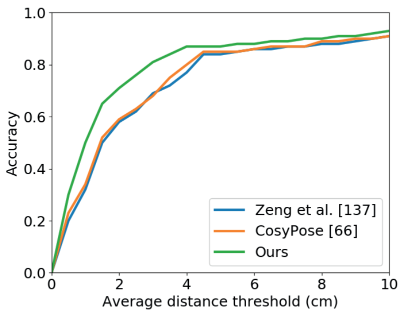
m is the number of points. For objects with symmetric views, we adapt the metric by computing the average distance using the closest point distance following prior works [10].

$$\mathrm{ADD-S} = \frac{1}{m}\sum_{x_1 \in M} \| \min_{x_2 \in M} (\bar{R}x_1 + \bar{T}) - (\hat{R}x_2 + \hat{T}) \| \tag{4.12}$$

Then we can compute the accuracy of predictions in average precision (AP) where a 6D pose estimate is considered to be true positive if the average distance is smaller than a fixed threshold [10]. However, this evaluation cannot reveal how an approach performs on these incorrect poses with respect to that threshold. Therefore, Xiang et al. [133] vary the distance threshold and plot an accuracy-threshold curve to compute the area under the curve for pose evaluation. We report the area under the accuracy-threshold curve (AUC) for 6D pose estimation on the YCB-Video dataset and warehouse dataset following PoseCNN [133] and DenseFusion [121]. The maximum threshold is set to 10cm. We also set the maximum threshold of AUC to be 2cm which is a lot more reasonable for most of the robot grippers.



(a) YCB objects



(b) Warehouse objects

Figure 4.12: Accuracy-threshold curves for all the 21 objects in the YCB-Video dataset and 11 objects in the warehouse dataset.

We compare our method with two multi-view-based approaches from the current state of the art [138, 66]. Fig 4.12 shows accuracy-threshold curves of the evaluated methods. Table 4.3 and 4.4 present a detailed evaluation for all the 21 objects in the YCB-Video dataset and 11 objects in the warehouse dataset. Our results show significant improvement in all objects by effectively employing more accurate projected mask, depth and color images from the high-quality semantic map. In addition, we validate the effectiveness of combining multiple views and the use of semantic map in object pose estimation by

Table 4.3: Area under the accuracy-threshold curve for 6D object pose estimation on the YCB-Video dataset. We compare our proposed system with multi-view-based methods [138, 66] that achieve state-of-the-art results on this dataset. We report the area under the accuracy-threshold curve (AUC). The maximum threshold is set to 10cm. We also set the maximum threshold of AUC to be 2cm which is a lot more reasonable for most of the robot grippers.

	[138]		[66]		Ours	
	10cm	2cm	10cm	2cm	10cm	2cm
002_master_chef_can	96.4	75.2	97.2	76.7	**97.6**	**80.1**
003_cracker_box	96.5	74.3	97.0	77.8	**97.7**	**81.3**
004_sugar_box	97.5	77.8	97.7	76.6	**98.4**	**82.0**
005_tomato_soup_can	94.6	71.0	95.9	75.4	**97.3**	**80.2**
006_mustard_bottle	97.2	76.8	98.1	76.9	**98.5**	**82.5**
007_tuna_fish_can	97.6	76.1	98.2	79.3	**98.7**	**81.0**
008_pudding_box	98.5	78.5	98.1	77.4	**98.4**	**79.9**
009_gelatin_box	98.1	81.1	98.5	80.0	**99.3**	**85.7**
010_potted_meat_can	94.3	72.0	94.5	74.5	**95.6**	**80.0**
011_banana	97.0	75.1	97.4	76.3	**98.2**	**81.1**
019_pitcher_base	98.5	77.0	99.2	78.6	**99.4**	**84.4**
021_bleach_cleanser	97.1	75.5	97.3	77.3	**98.0**	**81.2**
024_bowl	93.2	70.1	93.1	69.8	**95.6**	**81.4**
025_mug	98.2	81.4	98.6	80.0	**99.2**	**86.5**
035_power_drill	98.0	76.4	98.0	77.3	**98.1**	**79.6**
036_wood_block	94.7	73.7	95.3	75.2	**96.3**	**80.0**
037_scissors	97.2	78.0	97.3	76.5	**98.1**	**80.5**
040_large_marker	98.0	76.1	98.0	77.4	**98.8**	**82.5**
051_large_clamp	81.9	65.0	82.3	60.5	**85.2**	**75.3**
052_extra_large_clamp	77.9	65.9	78.2	64.4	**80.1**	**73.1**
061_foam_brick	94.5	71.1	94.9	70.5	**96.4**	**85.4**
MEAN	95.0	74.7	95.5	75.2	**96.4**	**81.1**

comparing it with the baseline DenseFusion as shown in Table 4.5. Our proposed object pose estimation system with semantic mapping achieves superior performance compared to the baseline single frame predictions. We observe that on both datasets combining information from multiple views improved the accuracy of the pose estimation over the original DensFusion. We see an improvement of 3.4% over the baseline single frame method with the proposed method, from 93.0% to 96.4% for the YCB-Video dataset. We also observe a marked improvement, from 60.5% for a single frame to 77.6% with our approach on the warehouse object dataset. Furthermore, we ran an ablations to

Table 4.4: Area under the accuracy-threshold curve (AUC) for the warehouse dataset. We compare our proposed method against the related works [138, 66]. We report the area under the accuracy-threshold curve (AUC). The maximum threshold is set to 10cm. We also set the maximum threshold of AUC to be 2cm which is a lot more reasonable for most of the robot grippers.

	[138]		[66]		Ours	
	10cm	2cm	10cm	2cm	10cm	2cm
001_frasvaf_box	70.0	57.5	69.9	57.1	**70.4**	**60.3**
002_small_jacky box	71.1	57.9	71.8	58.5	**75.6**	**64.8**
003_jacky_box	70.3	56.0	74.2	60.7	**76.7**	**66.5**
004_skansk_can	73.3	59.6	70.2	57.0	**74.8**	**63.3**
005_sotstark_can	69.6	56.3	70.3	56.7	**78.2**	**67.0**
006_onos_can	70.1	56.0	71.4	57.8	**80.0**	**69.3**
007_risi_frutti_box	69.9	55.1	68.5	54.5	**75.9**	**65.2**
008_pauluns_box	68.6	55.1	71.6	56.3	**82.2**	**67.7**
009_tomatpure	73.5	58.4	74.1	60.5	**83.5**	**65.5**
010_pallet	72.3	56.6	68.4	55.0	**77.1**	**62.5**
011_half_pallet	68.9	54.4	69.5	56.7	**78.5**	**61.9**
MEAN	70.7	56.6	70.9	57.3	**77.6**	**65.0**

Table 4.5: Ablation studies. We ran a number of ablations to analyze the contribution of different components to the performance of 6D object pose estimation including: DenseFusion using projected masks (DF-PM); DenseFusion using projected masks and projected depth (DF-PM-PD); DenseFusion using projected masks, depth, and RGB image (DF-PM-PD-PC); Our multi-view-based method without using projected masks, depth, and RGB images (Ours$^-$); Our full system (Ours). We report the area under the accuracy-threshold curve (AUC) with the maximum threshold is set to 10cm.

	YCB-Video Dataset	Warehouse Dataset
DF [121] (single-view)	93.0	60.5
DF-PM (single-view)	93.7	64.6
DF-PM-PD (single-view)	94.3	67.6
DF-PM-PD-PC (single-view)	94.5	68.8
Ours$^-$ (multi-view)	94.8	69.7
Ours (multi-view)	**96.4**	**77.6**

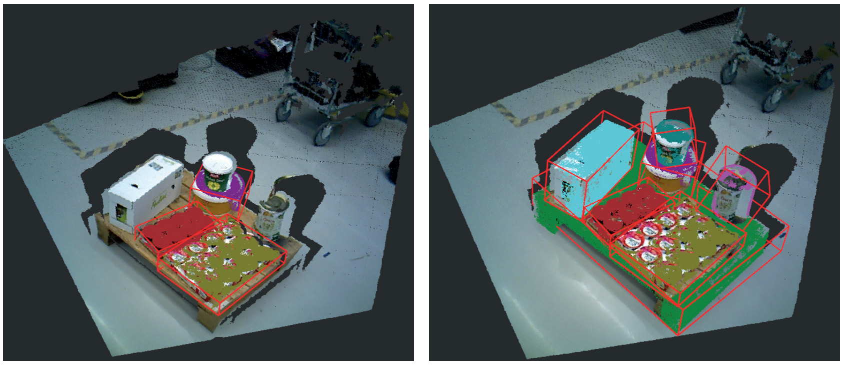
analyze the developed system including (i) DenseFusion using projected masks (DF-PM) (ii) DenseFusion using projected masks and projected depth (DF-PM-

Table 4.6: Average run-time analysis of system components (ms per frame).

Component	Run-time (ms)
Segmentation	50
Registration	30
Data Fusion	20
Object Pose Estimation	40
Total	140

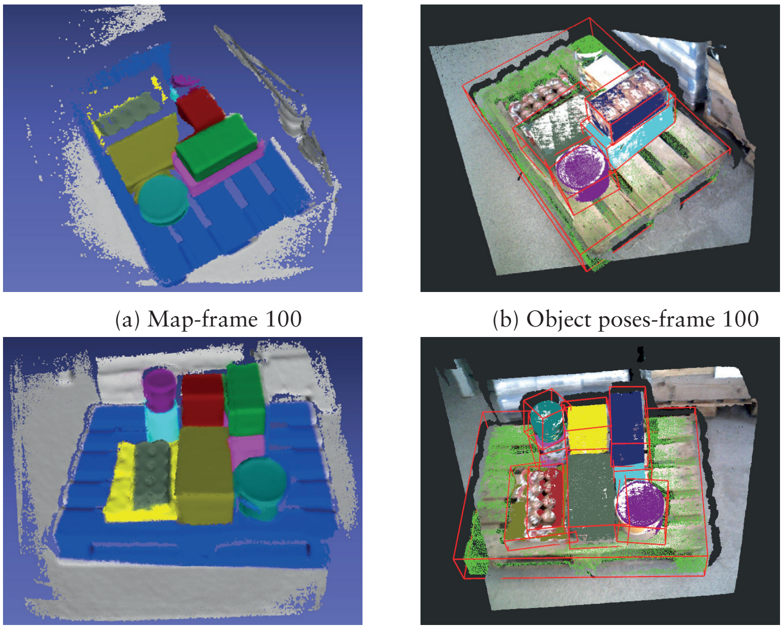
PD) (iii) DenseFusion using projected masks, projected depth, and projected RGB image (DF-PM-PD-PC). DF-PM performed better than DenseFusion on both datasets (+0.7% and +4.1%). For DF-PM-PD, performance improved additionally with +0.6% on the YCB-Video dataset and +3.0% on the warehouse object dataset. The performance benefit of DF-PM-PD-PC was less clear as it resulted in a smaller improvement of +0.2% and +1.2% over DF-PM-PD. The remaining improvement is due to the fusion of estimates in the EKF. Fig 4.13 and Fig. 4.14 visualize dense 3D semantic mapping and object pose estimation from the proposed system on 2 different videos in our warehouse dataset.

Lastly, the execution times of the individual components, averaged over all evaluated sequences, are shown in Table 4.6. Note that the segmentation computation time is a sum of semantic segmentation (15ms per frame) and instance segmentation (350ms per keyframe, 1 keyframe per 10 frames).



(a) Object poses-frame 50 (b) Object poses-frame 300

Figure 4.13: Object pose estimation from the proposed system on a video in the warehouse dataset.


(a) Map-frame 100 (b) Object poses-frame 100

(c) Map-frame 500 (d) Object poses-frame 500

Figure 4.14: Dense 3D semantic mapping and object pose estimation from the proposed system on a video in the warehouse dataset.

4.4.4 Robotic Manipulation Experiments

Our perception system has been integrated into a dual-arm robotic platform to perform picking operations for logistics in a completely autonomous manner [91]. An Asus Xtion PRO camera is mounted on top of a hand of the dual-arm manipulator as shown in Fig. 4.15. An extensive set of experiments on different objects and conditions is perfromed to assess the performance of the integrated system. A representative set of objects in the warehouse dataset was chosen for this evaluation. The particular object instances are motivated by the use case of the ILIAD EU project[3]. A task executed correctly consists of a correct detection of the objects to pick and a successful picking action. From this definition, the success rate of the platform can be expressed by the product of the detection rate and the picking rate retrieved from the experiments. The results are reported in Table 4.7.

[3]`https://iliad-project.eu/`

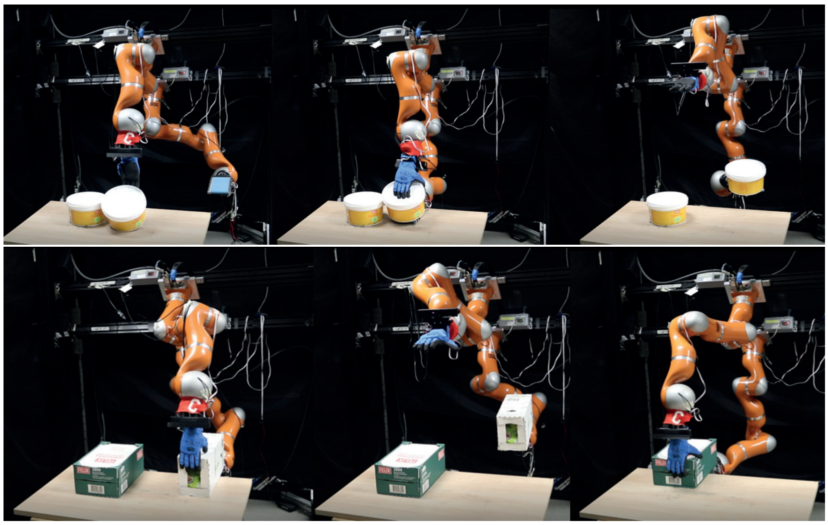


Figure 4.15: The developed object-oriented semantic reconstruction has been employed to perform robotic manipulation in Research Center E.Piaggio, University of Pisa [91].

Table 4.7: Performance of the integrated system for picking operations.

Objects	Detection rate	Picking rate	Success rate
Skansk	100%	100%	100%
Sotstark	90%	80%	72%
Jacky	100%	100%	100%
Tomatpure	100%	90%	90%
Pauluns	90%	100%	90%

4.5 Discussion

In this chapter, we have presented and validated a multi-view object pose estimation system with semantic mapping. The developed system yields high-quality semantic reconstruction while simultaneously recovering 6D poses of object instances. The main contribution of this chapter is to show that by taking advantage of deep learning-based techniques and our reconstruction system we are able to improve the performance of object pose estimation as compared to single view-based methods. We have provided an extensive evaluation on common benchmarks and our own dataset. The results confirm that the proposed method is able to produce a high quality dense map with robust tracking.

We also demonstrated that the proposed object pose estimator benefits from the use of mask, depth and RGB images generated by the mapping system and from combining multiple predictions based on the Kalman filter. We believe that accurate scene reconstruction and object pose estimation from multi-views will open the way to new applications regarding autonomous robotic manipulation.

In this chapter, we present systems capable of reconstructing highly detailed object-level models and estimating the 6D pose of objects by means of an RGB-D camera. However, these methods require large amounts of labeled training data. Collecting images of objects from the real world under various conditions and annotating the images with 6D object poses is time-consuming and requires a significant human effort. In addition, in a number of cases, color information may not be available, such as measured point cloud data from laser range finders or industrial high-resolution 3D sensors. In the next chapter, we present a method for recovering the 6D pose of rigid objects from 3D point clouds containing only geometric information and using only synthetic data for training.

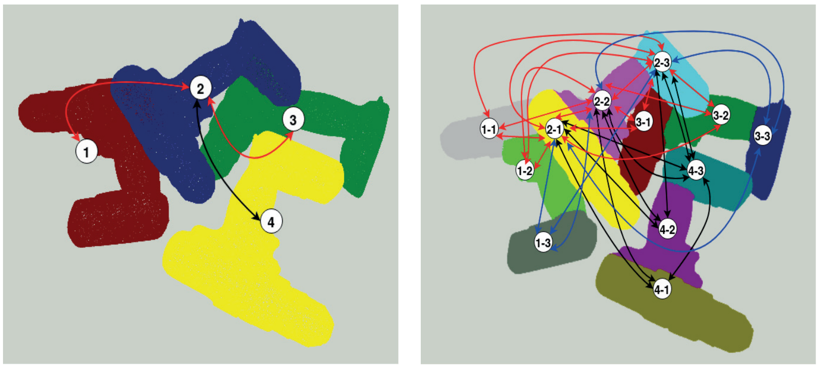
Chapter 5
Object Pose Estimation from 3D Point Clouds

5.1 Introduction

In Chapter 4, by taking RGB-D images as input we have achieved state-of-the-art performance on the object pose estimation task. However, in a number of cases, color information may not be available — for example, when the input is point cloud data from laser range finders or industrial high-resolution 3D sensors. Therefore, studies on recovering the 6D pose of rigid objects from 3D point clouds containing only geometric information are necessary.

Deep learning-based methods require large amounts of labeled training data. Collecting images of objects from the real world under various conditions and annotating the images with 6D object poses is time-consuming and requires a significant human effort. A promising alternative is the use of synthetic data for training such deep neural networks. Nonetheless, it is still difficult for methods requiring color information where the domain gap between synthetic training and real test images is severe. Compared to color images, the domain gap between the synthetic and real data is considerably smaller for 3D point clouds [50, 77]. In addition, synthesizing data with only geometric information is less expensive in the terms of time and hardware storage as there is no texture or illumination present in the data. This low cost allows us to scale it up to a large number of objects, which is often desired in practical applications. This ability to generate vast quantities of data, coupled with the relatively small domain gap between simulated and real data, suggests that it is feasible to develop DNN-based object pose estimation methods that can be trained purely in simulation and work well in the real world.

In this chapter, we propose a novel DNN architecture to address the problem of estimating the 6D pose of multiple rigid objects in a cluttered scene, using only a 3D point cloud of the scene as an input. We build our network on top of a deep Hough voting architecture [97], VoteNet. The voting mechanism allows



(a) Object instances (b) Object parts

Figure 5.1: An example to illustrate the pose relation between object parts and object instances. The red, blue, and black arrows indicate high, fair, and low correlations respectively.

our model to perform reliable detection under clutter and occlusion. More importantly, we explore the relation and contextual information between objects (instances) and local parts of objects (parts), which are crucial for the 6D pose estimation task as shown in Fig. 5.1. To this end, we introduce two high-level feature learning modules ($\mathtt{M}_{\mathtt{O}}$ and $\mathtt{M}_{\mathtt{P}}$) into the VoteNet architecture to model the pose relation between object instances and object parts in the scene. The module $\mathtt{M}_{\mathtt{O}}$ first generates object-centric proposals using a per-point voting scheme inspired by VoteNet. We then feed proposal features into a self-attention module in order to enable higher-order interactions between neighboring proposal features and learn instance-to-instance correlations. Similarly, the module $\mathtt{M}_{\mathtt{P}}$ votes for object part centers and then learns part-to-part correlations with self-attention.

The main contributions of this chapter are: (1) An end-to-end trainable network for 6D object pose estimation from 3D point clouds: robust to noise and occlusion, and able to deal with multiple objects in cluttered scenes; (2) Two high-level feature learning modules for modeling part-to-part and instance-to-instance correlations with self-attention to improve the performance of object pose estimation; (3) Extensive experiments demonstrating the benefits of high-level feature learning.

5.2 Hough Voting in Computer Vision

The Hough Transform has been widely used in computer vision for tasks like object detection [108, 117], motion detection [59], medical imaging [35], and robot navigation [58] for decades. It was originally introduced to detect analytically defined shapes such as line, circle or ellipse [52, 53, 27] in 2D images. Today, Hough Transform or Hough voting usually refers to any detection process where evidence coming from local elements is accumulated to form a confident detection. Voting-based approaches [117, 97] demonstrated the ability to perform reliable detection under clutter and occlusion. This is due to the additive attribute of the Hough transform which makes the method robust to partial occlusions. Tombari and Di Stefano [117] proposed a Hough Voting approach for object recognition in 3D scenes. The key technique of their method is that each corresponding feature can cast a vote to accumulate evidence for possible object centers. This permits simultaneous voting of all feature correspondences in 3D Hough space. In [61], researchers presented a 3D object detection and pose estimation method by combining neural networks and a local voting-based approach.

Recently, VoteNet [97] was introduced to detect objects via point feature grouping, sampling, and voting. While traditional Hough voting is difficult to optimize jointly due to multiple separate modules, VoteNet is end-to-end optimizable. Fig. 5.2 illustrates the architecture of this end-to-end detection network. VoteNet directly votes for center points of objects from point clouds and generates a group of high-quality 3D object proposals by aggregating vote features. The experimental results show that the developed methods above perform well in 3D scenes with a significant degree of occlusion and clutter. The output of VoteNet and other 3D object detection approaches is 3D bounding boxes as well as semantic classes of objects. In robot manipulation, rather than proposing bounding boxes, we are more interested in estimating the 6D pose of objects as discussed in Chapter 2. Motivated by the success of VoteNet in the object detection task, in this chapter we build a 6D object pose estimation network upon the deep Hough voting module. However, extending VoteNet for object pose estimation task is not straightforward. Although VoteNet had achieved favorable performance on 3D object detection, it fails to explore the correlation between objects (instances) and local parts of objects (parts), which are crucial for the 6D pose estimation task. The correlations do not relate to semantic meanings, but rather allow the propagation of information about the spatial relationship between objects and parts. For instance, in Fig. 5.1 we have contextual information: part 2-1 is on top of part 1-1, part 2-3 is on top of part 3-2, part 2-2 connects part 2- 3, and part 2-1. From these relationships, we can infer that object 2 locates somewhere in between object 1 and object 3 and does not lie on the supporting surface or standing. Therefore, in Section 5.4 we propose self-attention modules for encoding the dependency of object parts and object instances into features to boost the performance of object pose estimation.

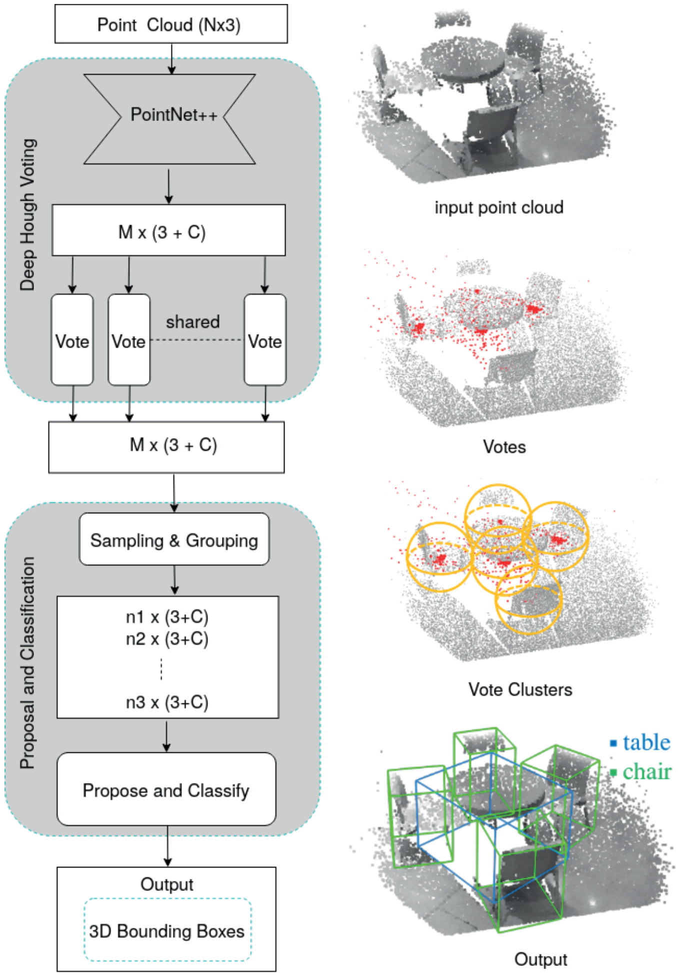


Figure 5.2: Architecture details of VoteNet [97] for 3D object detection in point clouds. Given an input point cloud, PointNet++ is used as a backbone network to subsamples and learns features and outputs M seed points of dimension (3 + C). This subset of points are considered as seed points. Then each seed point generates one vote. In the next step, the votes are grouped into clusters and processed by the proposal module to generate 3D bounding boxes.

5.3 Self-Attention

Attention mechanisms allow deep learning models to learn to focus on important regions within a context and have been widely applied in a variety of tasks [82, 5, 17]. Self-attention is an attention mechanism that allows inputs to interact with each other and to learn the correlation between them. The ability of self-attention to directly model long-distance interactions and to reduce sequential computation have revolutionized machine translation and natural language processing [119, 24]. This has inspired applications of self-attention to computer vision tasks such as image recognition [54], semantic segmentation [31], and image captioning [135]. As discussed in [123], self-attention can be viewed as a form of the non-local mean [11]. The non-local module [123] is designed to capture long-range spatio-temporal dependencies in images and videos. It can be integrated into many computer vision architectures. The authors demonstrated that it can greatly improve the performances of existing deep neural networks on many benchmarks for video classification and static image recognition tasks. Following [123], a non-local operation in deep neural networks is defined as:

$$Y = f(\theta(X), \phi(X))g(X) \tag{5.1}$$

where $X \in \mathbb{R}^{N\times C}$ is the input feature map, Y is the output signal of the same size as X, C is the number of channels, and N is the number of positions $N = HW$ (width W and height H). $\theta(\cdot), \phi(\cdot), g(\cdot)$ are parametrized function approximators on the input. Using 1×1 convolution, the transformations can be written as:

$$\theta(\cdot) : \mathbb{R}^{N\times C} \times \mathbb{R}^{C\times C} \to \mathbb{R}^{N\times C},\ \theta(X) = XW_\theta \in \mathbb{R}^{N\times C} \tag{5.2}$$

$$\phi(\cdot) : \mathbb{R}^{N\times C} \times \mathbb{R}^{C\times C} \to \mathbb{R}^{N\times C},\ \phi(X) = XW_\phi \in \mathbb{R}^{N\times C} \tag{5.3}$$

$$g(\cdot) : \mathbb{R}^{N\times C} \times \mathbb{R}^{C\times C} \to \mathbb{R}^{N\times C},\ g(X) = XW_g \in \mathbb{R}^{N\times C} \tag{5.4}$$

parameterized by the weight matrices $W_\theta, W_\phi, W_g \in \mathbb{R}^{C\times C}$ respectively. The function $f(\cdot,\cdot) : \mathbb{R}^{N\times C} \times \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times N}$ computes the affinity between all positions. The non-local behavior in Eq. 5.1 is due to the fact that all positions are considered in the operation. There are various choices for f. [123] describes several versions of f including Gaussian function, embedded Gaussian, dot-product, and concatenation. Dot-product is considered as the simplest one:

$$f(\theta(X), \phi(X)) = \theta(X)\phi(X)^\top \tag{5.5}$$

The non-local operation (Eq. 5.1) captures dependencies between any two positions via a joint location-wise matrix $f(\theta(X), \phi(X))$ by aggregating all channel

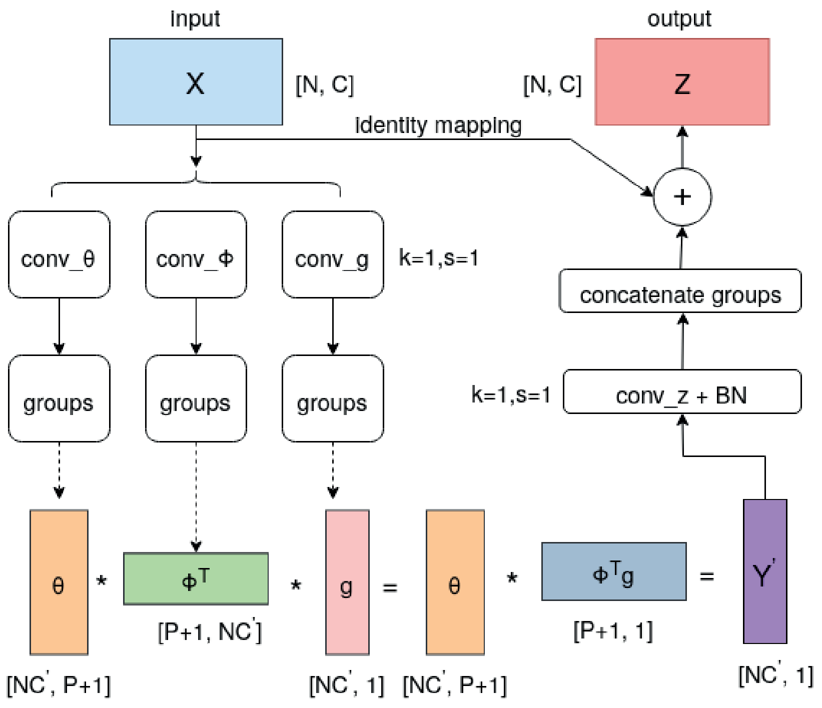


Figure 5.3: Compact generalized non-local (CGNL) module [137]. The input feature map X of the size (N $\times$ C) is first fed into three 1 $\times$ 1 $\times$ 1 conv layers and then divided along channels into multiple groups. The channels dimension is grouped into $C' = C/G$, where G is a group number. The compact representations for generalized non-local module are build within each group. P and BN denotes the order of Taylor expansion for kernel functions and BatchNorm respectively.

information together. The Compact Generalized Non-local Network (CGNL) [137] (Fig. 5.3) explores channel-wise correlations to generalize the non-local module and learn explicit correlations among all of the elements across channels. According to the experimental results in [137], CGNL demonstrated the generality of non-local operations and can become a basic building block in designing deep neural networks.

5.4 Object Pose Estimation in 3D Point Clouds

Conventional methods [105, 131, 2, 40, 26] are mainly based on the matching of local or global features extracted from input point cloud to features in a 3D model of the object. Global feature-based approaches utilize the whole geometric appearance of the object surface to define a single feature vector that effectively and concisely describes the entire 3D object [105, 131, 2]. On the contrary, the local feature-based methods exploit the geometric properties around specific keypoints [40]. While global methods are able to handle objects with self-similar surface parts, such as planar patches, spheres and cylinders, local approaches are more suitable for detecting and estimating the pose of complex objects in cluttered scenes. Drost et al. [26] attempt to combine the advantages of the two by using point pair features and match that model locally using a fast voting scheme. The proposed point pair feature (PPF) describes the relative position and orientation of two oriented points. The algorithm was shown to achieve high scores on standard object recognition benchmarks [10] and is utilized in a vast number of industrial and robotic applications [16, 122]. Since introduced, PPF has been improved and extended in many other works [7, 49, 39, 70]. Although extensively studied, PPF-based approaches share some common problems: (1) relying on searching a large set of paired feature correspondences severely limits their speed; (2) they are sensitive to measurement noise, heavy occlusion and background clutter. Recent work has attempted to leverage the power of deep learning to achieve fast and robust 6D pose estimation [121, 46, 14, 42, 33]. However, these approaches either still require color information in the process or rely on a prior instance segmentation step. As the quality of the segmentation mask greatly influences the performance of pose estimation, these approaches are often limited by poor performance of segmentation in the presence of clutter and occlusion. Unlike existing methods, our model consumes unordered 3D point cloud data without color information and does not require segmentation masks. Our method relies on a voting mechanism to perform reliable detection and learns the dependency of object parts and object instances to boost the performance of pose estimation. The details of our developed model are presented in the rest of this section.

5.4.1 Overview

The proposed network is illustrated in Fig. 5.4 and Fig. 5.5. The architecture comprises of four components. The first one takes as input a 3D point cloud and extracts seeds as high-dimensional features. The second component $\mathtt{M_P}$ operates on the high-dimensional features to learn part proposals and uses a self-attention module to enable higher-level interactions between part proposals. Similarly, the third one $\mathtt{M_O}$ takes seeds as input to learn object proposals and then encode the instance-to-instance correlation information through a self-attention module. The voting part in both $\mathtt{M_P}$ and $\mathtt{M_O}$ extend the idea about deep Hough voting in VoteNet [97] to estimate centers of object parts and object instances. Last, a pose estimation module combines feature maps from the self-attention modules and generates the pose of objects.

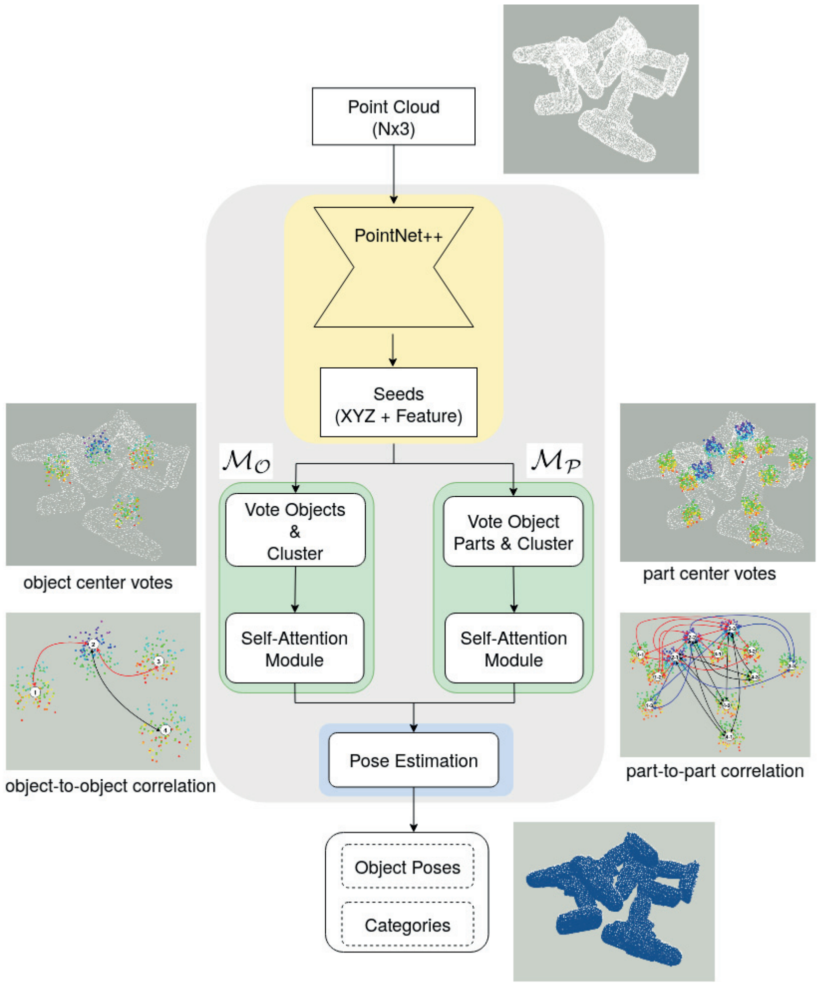


Figure 5.4: The architecture of the proposed network for 6-DOF object pose estimation in point cloud data. Our model builds on the deep Hough voting module of VoteNet [97] to vote objects and parts. Especially, we introduce self-attention modules for encoding the dependency of object parts and object instances into features to boost the performance of object pose estimation.

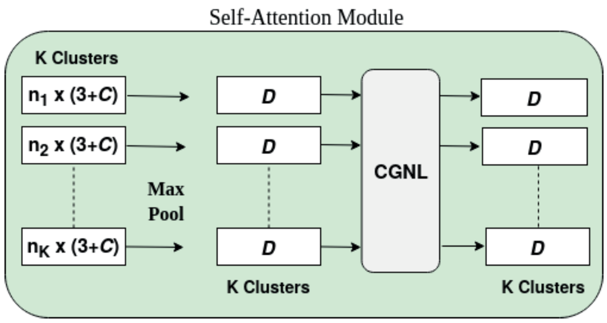


Figure 5.5: Self-attention module with generaized non-local network (CGNL) [137].

5.4.2 Feature Extraction

In order to extract geometric features, we utilize the PointNet++ architecture with multi-scale grouping as our backbone network. As the core of our base network, its set abstraction level is composed of three layers: a sampling layer, a grouping layer, and the PointNet [98] based learning layer. Through PointNet++, we are able to capture fine geometric structures from the neighborhood of each point. The base network selects M interest points (called seed points) and enriches them with high-dimensional features $\{s_i\}_{i=1}^{M}$ where $s_i = [x_i; f_i]$ with $x_i \in \mathbb{R}^3$ being the seed location in 3D space and $f_i \in \mathbb{R}^C$ being a feature vector.

5.4.3 Learning Part-to-Part Correlation

Given the high-dimensional features $\{s_i\}_{i=1}^{M}$, a part detection module $\mathcal{M}_{\mathcal{P}}$ is used to generates a fixed number J of proposals. A proposal is a tuple (z_i, h_i, s_i) consisting of a position $z_i \in \mathbb{R}^3$, a proposal features vector $h_i \in \mathbb{R}^D$ and a set of points s_i associated with the proposal. To be specific, each seed point s_i is fed into a shared Multi-Layer Perceptron (MLP) to compute a feature offset $\Delta f_i \in \mathbb{R}^C$ and a relative 3D offset $\Delta x_i \in \mathbb{R}^3$ between the point position $x_i \in \mathbb{R}^3$ and its corresponding part center $c_i \in \mathbb{R}^3$. Then the vote can be denoted as $v_i = [y_i; g_i] \in \mathbb{R}^{3+C}$ with $y_i = x_i + \Delta x_i$ and $g_i = f_i + \Delta f_i$. To supervise the learning of the 3D offset Δx_i, we apply a regression loss:

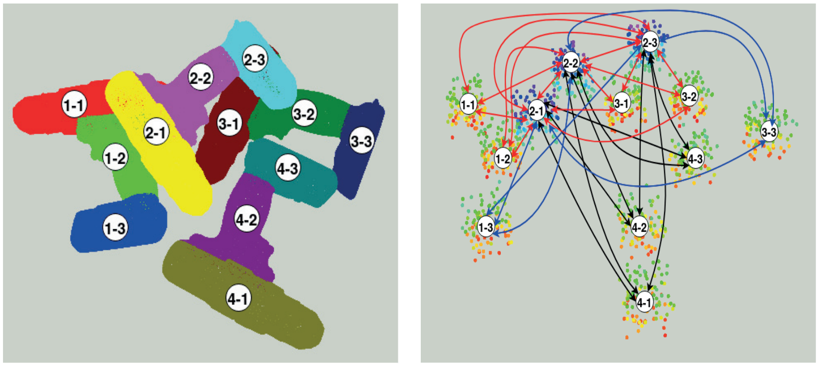
$$L_{part-vote} = \frac{1}{M_{pos}} \sum_i \|x_i + \Delta x_i - c_i^p\|_H \cdot \mathbb{1}(x_i) \qquad (5.6)$$

where M_{pos} is the count of the total number of seeds on the object surface, $\|\cdot\|_H$ is the Huber norm and $\mathbb{1}(\cdot)$ is a binary function indicating whether a seed point s_i belongs to an object part. After each seed point has voted for a part center, we obtain a distribution over object part centers. The next step is to form J vote clusters by uniform sampling and finding neighboring votes within a certain Euclidean distance. Then votes from J clusters are aggregated to generate J feature proposals $\{h_i \in \mathbb{R}^D\}_{i=1}^J$ using a PointNet-like module as described in [97]. At this stage, we have J proposals composed of 3D positions $z_i = x_i + \Delta x_i$ located near part centers, proposal features $h_i \in \mathbb{R}^D$ describing the local geometry, and a set of seed points s_i associated with each proposal.

So far, the proposal feature maps $H = \{h_i\}_{i=1}^J$ encode only local information of the point cloud. In order to enable features to become aware of their global neighborhood, we explicitly model higher-order interactions between proposal features, and it can be formulated as the non-local operation:

$$H_{part-part} = f(\theta(H)\phi(H))g(H) \tag{5.7}$$

where $\theta(\cdot)$, $\phi(\cdot)$, $g(\cdot)$ are learnable transformations on the input feature map H, and $f(\cdot)$ encodes the relation between any two parts. To this end, we opt for the compact generalized non-local network (CGNL) [137] as our self-attention module to explicitly model rich correlations between parts and to provide higher-level feature learning in addition to the lower-level point features. Fig 5.6 illustrates object part center votes and relation between parts.



(a) (b)

Figure 5.6: An example to illustrate learning in object part center votes and seft-attention module. The red, blue and black arrows indicate high, fair and low correlations respectively.

5.4.4 Learning Instance-to-Instance Correlation

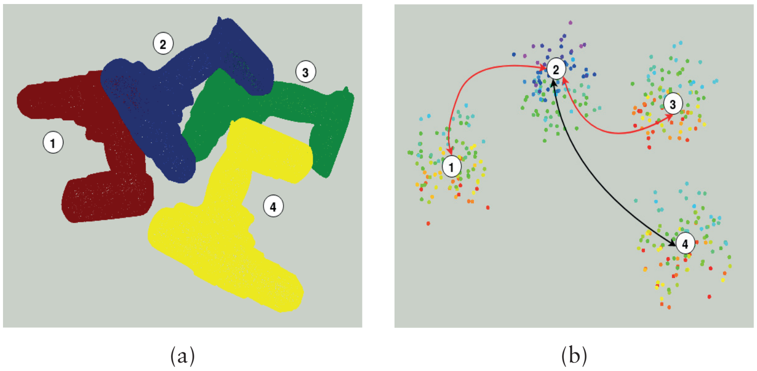


Figure 5.7: An example to illustrate learning in object center votes and self-attention module. The red, blue and black arrows indicate high, fair and low correlations respectively.

Similarly, an object detection module $\mathcal{M}_{\mathcal{O}}$ is used to generate K clusters from the high-dimensional features $\{s_i\}_{i=1}^{M}$ and a set of object centers. The loss is defined as:

$$L_{object-vote} = \frac{1}{M_{pos}} \sum_{i} \|x_i + \Delta x_i - c_i^o\|_H \cdot \mathbb{1}(x_i) \tag{5.8}$$

where Δx_i is a Euclidean space offset between the point position x_i and its corresponding ground-truth object center c_i^o. To encode the relationship of objects in the scene into features or exploit information outside of local regions, instead of processing each cluster independently, our network computes a new feature map from all clusters to learn higher-level features that consider the relationships between all object instances. Fig 5.7 illustrates object center votes and relation between object instances. By leveraging a self-attention mechanism, we can combine features from clusters to enable higher-order interaction between proposals. We thus make use of the compact generalized non-local network (CGNL) [137]. CGNL allows for explicit modeling of rich inter-dependencies between clusters in feature space in a fast and low-complexity computation flow. The CGNL based self-attention module takes K clusters $C = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_K\}$ as inputs. Then votes from each cluster are processed by a MLP before being max-pooled to a single feature vector and passed to CGNL. The self-attention mechanism allows the features from different clusters to interact with each other. The output is a new feature map $H_{obj-obj} = \{H^k_{obj-obj}\}$ with $k = 1, ..., K$.

$$H_{obj-obj} = \text{CGNL}(\max_{i=1,\ldots n}(\text{MLP}(v_i))) \tag{5.9}$$

5.4.5 Multi-task Loss

Given new feature maps $H_{part-part}$ and $H_{obj-obj}$ generated by the self-attention modules, max-pooling is first applied to get two vectors including the part feature vector and the object feature vector, combining information from all the part and object candidates. These two vectors are then concatenated to form a single feature vector. An MLP layer is applied to further aggregate global information. While we can learn the translation part of the pose in the Euclidean space, representing the rotation part is more complicated. The common options of using Euler angles-based and quaternion-based representations are discontinuous and difficult for neural networks to learn as explained by [142]. In [142], the authors show that the 3D rotations have continuous representations in 5D and 6D, which are more suitable for learning. Therefore, we make use of 6D continuous representation for 3D rotations. Following [142], we map the 6D representation produced by the network into the original rotation space and minimize the L2 loss between the output and the ground-truth rotation matrices.

We supervise the learning of modules jointly with a multi-tasks loss:

$$L = \lambda_1 L_{part-vote} + \lambda_2 L_{obj-vote} + \lambda_3 L_{pose} \tag{5.10}$$

where λ_1, λ_2 and λ_3 are the weights for each task. The loss includes a voting part loss $L_{part-vote}$, a object vote loss $L_{obj-vote}$, and a pose loss L_{pose}. We define the pose loss function as follows:

$$L_{pose} = L_t + \alpha L_{rot} + \beta L_{obj} + \gamma L_{sem} \tag{5.11}$$

where α, β, and γ are weights that scale the losses to similar scales. The pose loss is composed of a translation loss L_t (regression), an L2 loss between the output and the ground-truth rotation matrices, an objectness loss L_{obj} and a semantic classification loss L_{sem}. The objectness loss is a cross-entropy loss for two classes (an object or not). The semantic classification loss is also a cross-entropy loss of NC semantic classes. Hovever, the above loss L_{rot} for rotation can only appropriate to asymmetric objects. For symmetric objects having multiple correct 3D rotations, given the ground truth rotation $\bar{R}$ and translation $\bar{T}$ and the estimated rotation $\hat{R}$ and translation $\hat{T}$ we compute L_{rot} as:

$$L_{rot} = \frac{1}{m} \sum_{x_1 \in M} \| \min_{x_2 \in M} (\bar{R}x + \bar{T} - \hat{R}x + \hat{T}) \| \qquad (5.12)$$

where M denotes the set of 3D model points and m is the number of points. The loss is calculated as the average distance from vertices of the object model in the ground-truth pose to closest vertices of the model in the estimated pose. That way the loss is minimized when the two 3D models are aligned with each other.

5.4.6 Implementation Details

Table 5.1: Layer parameters of the PointNet++ [99] based feature learning network.

layer name	input layer	layer params
SA1	point cloud	(2048,0.025,[64,64,128])
SA2	SA1	(1024,0.05,[128,128,256])
SA3	SA2	(512,0.1,[128,128,256])
SA4	SA3	(256,0.2,[128,128,256])
FP1	SA3, SA4	[256,256]
FP2	SA2, SA3	[256,256]

Network Architecture. In our implementation we randomly choose N=50k points from each raw point cloud and set λ_1=0.5, λ_2=1.0, λ_3=0.1 in Eq. 5.10, α=β=γ=1.0 in Eq. 5.11. We then apply the PointNet++ [99] based feature learning network, which has 4 set abstraction layers (SA) and 2 feature propagation layers (FP). The detailed layer parameters are shown in Table 5.1. The FP2 outputs seeds that will be transformed to votes. The voting module generates $J = 2$ votes per seed with an MLP layer spec: [256, 256, 259 $\times$ 2], 1 vote for the object center and 1 vote for the part center. In the learning part-to-part and instance-to-instance correlation modules, we form 1024 clusters by finding neighboring votes and finally output a new feature map for each cluster. In the last step, 256 proposals are generated from 256 vote clusters sampled from the 1024 part-centric vote clusters and 1024 instance-centric vote clusters in the previous step.

Training the network. Our proposed model is trained from scratch in an end-to-end manner using the Adam optimizer. We train the entire network with a batch size of 48 and learning rate 0.001 for 200 epochs. It takes around 10 hours for training the Siléane and Fraunhofer IPA dataset [10, 64] using six

Nvidia Tesla V100 32GB GPUs. In regard to inference time, it takes around 150ms to process an input containing 50k points.

5.5 Evaluation

We evaluated our proposed approach on data from the Siléane dataset [10] and the Fraunhofer IPA dataset [64]. These datasets consist of multiple rigid texture-less objects of the same type under cluttered scenes with multiple and heavy occlusions. A comparison against the most closely related works is also performed here.

5.5.1 Datasets

The Siléane dataset [10] consists in a total of 2,601 independent scenes, fully annotated. Depending on the object, the dataset comprises 46 to 325 scenes and is too small to train our model. Therefore, we used this dataset only for testing. For model training, we relied on the Fraunhofer IPA Bin-Picking dataset [64]. It includes 520 fully annotated point clouds and corresponding depth images of real-world scenes (for testing) and about 206,000 synthetic scenes (for training). The dataset comprises eight objects from the Siléane dataset and two newly introduced ones (gear shaft and ring screw). Training data for the coffee cup (C.cup) was not generated in the Fraunhofer IPA dataset. Hence, we precisely rebuilt the setup from the dataset in simulation and produced 10,000 scene point clouds depicting various numbers of C.cup instances in bulk.

5.5.2 Evaluation metric

The error of an estimated pose $\hat{P}$ with respect to the ground-truth pose $\bar{P}$ of an object model M is measured by the most widely used pose-error function Average Distance of Model Points (ADD) [48]. The error is calculated as the average distance from vertices of the object model in the ground-truth pose to vertices of the model in the estimated pose. Given the ground truth rotation $\bar{R}$ and translation $\bar{T}$ and the estimated rotation $\hat{R}$ and translation $\hat{T}$, the average distance is defined as:

$$\mathrm{ADD} = \frac{1}{m} \sum_{x \in M} \| (\bar{R}x + \bar{T} - \hat{R}x + \hat{T}) \| \tag{5.13}$$

m is the number of points. For objects with symmetric views, we adapt the metric by computing the average distance using the closest point distance following prior works [10]. We report the accuracy of predictions in average precision (AP) following [10], given the goal of retrieval of instances less than 30% occluded. A 6D pose estimate is considered to be true positive if the error is smaller than 0.1 times the diameter of the smallest bounding sphere.

5.5.3 Results

Table 5.2: Results (AP) on non bin-picking scenarios. The methods includes the baseline (modified VoteNet [97] for 6D pose estimation), our proposed network algorithm without learning object-object correlations (Ours (-$\mathtt{M}_{\mathtt{O}}$)), without learning part-part correlations (Ours (-$\mathtt{M}_{\mathtt{P}}$)), with full options (Ours (full)).

	Baseline	Ours (- $\mathtt{M}_{\mathtt{O}}$)	Ours (- $\mathtt{M}_{\mathtt{P}}$)	Ours (full)
Brick	0.35	0.46	0.37	**0.48**
Bunny	0.49	0.63	0.51	**0.65**
C.stick	0.47	0.63	0.49	**0.66**
C.cup	0.39	0.54	0.42	**0.57**
Gear	0.56	0.64	0.57	**0.65**
Pepper	0.31	0.44	0.33	**0.45**
Tless 20	0.35	0.42	0.35	**0.42**
Tless 22	0.29	0.36	0.30	**0.38**
Tless 29	0.38	0.46	0.40	**0.46**
Gear shaft	0.58	0.65	0.60	**0.65**
Ring screw	0.60	0.67	0.62	**0.69**
MEAN	0.43	0.54	0.45	**0.55**

Table 5.3: Results (AP) on bin-picking scenarios.

	Baseline	Ours (- $\mathtt{M}_{\mathtt{O}}$)	Ours (- $\mathtt{M}_{\mathtt{P}}$)	Ours (full)
Brick	0.28	0.38	0.41	**0.47**
Bunny	0.32	0.48	0.49	**0.58**
C.stick	0.31	0.39	0.44	**0.55**
C.cup	0.25	0.36	0.39	**0.47**
Gear	0.45	0.58	0.60	**0.63**
Pepper	0.20	0.25	0.27	**0.34**
Tless 20	0.27	0.37	0.39	**0.45**
Tless 22	0.20	0.32	0.33	**0.36**
Tless 29	0.28	0.37	0.39	**0.45**
Gear shaft	0.41	0.57	0.58	**0.64**
Ring screw	0.43	0.55	0.57	**0.65**
MEAN	0.31	0.42	0.44	**0.51**

Table 5.2 and Table 5.3 present a detailed evaluation for 9 objects in the Siléane dataset [10] and 2 objects (gear shaft and ring screw) in the Fraunhofer

IPA dataset [64]. We modified the VoteNet architecture to directly estimate the 6D pose of objects from the vote aggregated features and used it as the baseline method. Our proposed model achieves superior performance compared to the modified VoteNet. We observe that in all cases learning part-to-part and object-to-object correlations improved the accuracy of the pose estimation over the baseline method. We see an improvement of +12% over the baseline method with our proposed model, from 43% to 54% for the non bin-picking scenarios. We also observe a marked improvement, from 31% to 51% on bin-picking scenarios. Fig 5.8 example of scenes in the Siléane dataset [10] and the Fraunhofer IPA dataset [64]. Fig. 5.9 and Fig. 5.10 demonstrate qualitative results.

Furthermore, we ran a number of ablations to analyze our network without learning part-to-part correlation (Ours (- $\mathtt{M_P}$)) and without learning object-to-object correlation (Ours (- $\mathtt{M_O}$)). Ours (- $\mathtt{M_O}$) performed better than the baseline method on both bin picking and non-bin picking scenarios (+11% and +11%). The performance benefit of (- $\mathtt{M_P}$) was less clear as it resulted in a small improvement of +2% on non bin-picking scenarios. On the contrary, performance improved significantly with +13% on the bin-picking scenarios. This suggests that our framework makes efficient use of the object-to-object correlation information to improve pose estimation.

Table 5.4: AP values of conventional algorithms and our method on objects from the Siléane dataset [10] and the Fraunhofer IPA dataset [64] (both non bin-picking and bin-picking scenarios).

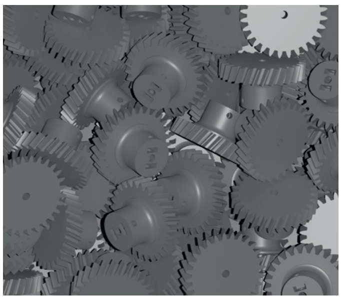
	[48]	[48]+PP [1]	PPF [26]	PPF+PP [1]	[70]	Ours
Brick	0.10	0.17	0.19	0.24	0.37	**0.48**
Bunny	0.38	0.42	0.27	0.34	0.44	**0.61**
C.stick	0.37	0.45	0.15	0.21	0.50	**0.60**
C.cup	0.08	0.19	0.27	0.28	0.40	**0.52**
Gear	0.21	0.26	0.28	0.31	0.40	**0.64**
Pepper	0.04	0.03	0.06	0.12	0.28	**0.39**
Tless 20	0.11	0.13	0.21	0.26	0.31	**0.44**
Tless 22	0.09	0.12	0.13	0.17	0.22	**0.37**
Tless 29	0.19	0.21	0.28	0.33	0.39	**0.46**
Gear shaft	0.18	0.20	0.32	0.34	0.38	**0.65**
Ring screw	0.27	0.33	0.40	0.46	0.54	**0.67**
MEAN	0.18	0.23	0.23	0.28	0.38	**0.53**

Table 5.4 and Table 5.5 summarize the comparison results between our method and current point cloud-based approaches on the same objects from above. The experiments are based on publicly available implementations by the authors [48, 14, 1, 33], and on our own implementation of the rest. Our

proposed approach achieves state-of-the-art results and outperforms others on all objects, obtaining an average precision of 53%. As can be seen from the obtained results, the proposed method results in an improvement of +15% to +35% on AP compared with the conventional approaches [48, 26, 1, 70]. The AP score of our method also surpasses the existing deep learning-based methods [14, 42, 33] with a significant margin.

Table 5.5: AP values of deep learning-based methods on objects from the Siléane dataset [10] and the Fraunhofer IPA dataset [64] (both non bin-picking and bin-picking scenarios).

	G2l-net [14]	[42]	[33]	Ours
Brick	0.42	0.36	0.38	**0.48**
Bunny	0.54	0.46	0.52	**0.61**
C.stick	0.54	0.50	0.52	**0.60**
C.cup	0.45	0.41	0.44	**0.52**
Gear	0.51	0.42	0.50	**0.64**
Pepper	0.30	0.24	0.25	**0.39**
Tless 20	0.38	0.35	0.30	**0.44**
Tless 22	0.29	0.20	0.23	**0.37**
Tless 29	0.35	0.40	0.37	**0.46**
Gear shaft	0.48	0.41	0.51	**0.65**
Ring screw	0.56	0.45	0.54	**0.67**
MEAN	0.44	0.38	0.41	**0.53**


(a)

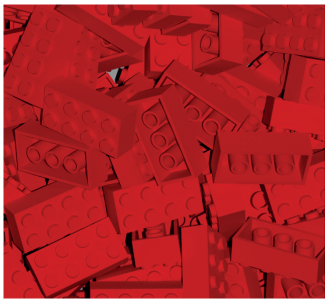

(b)

Figure 5.8: Example of scenes in the Siléane dataset [10] and the Fraunhofer IPA dataset [64].

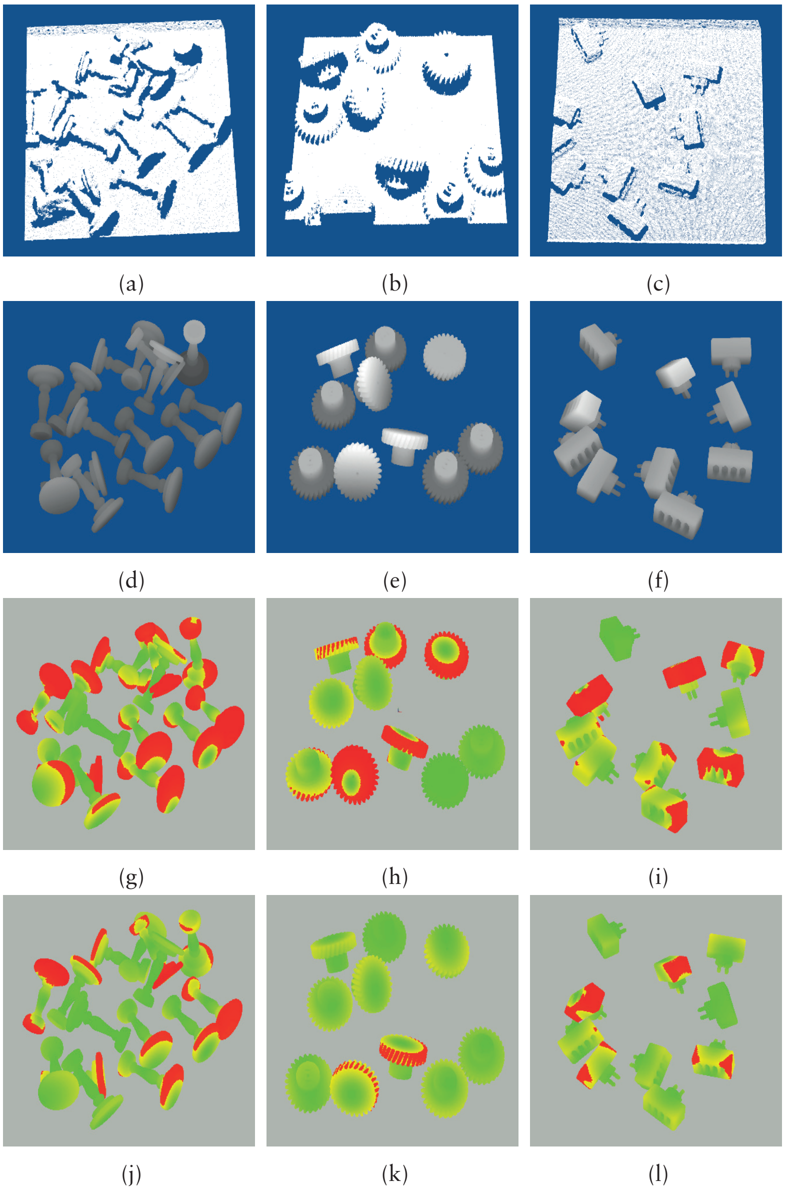
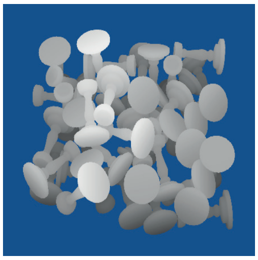


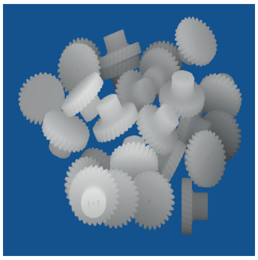
Figure 5.9: Visualization for pose estimation results on non bin-picking scenarios: (a-c) 3D point cloud inputs; (d-f) ground truth poses; (g-i) retrieved pose hypotheses by the baseline method; (j-l) retrieved pose hypotheses by our proposed method. Color-coded visualization of point-wise distance error ranging from 0 (green) to greater than 0.2 times the diameter of the object (red). Predicted poses with ADD more than 0.2 times the diameter of the object have been removed for visualization purposes.

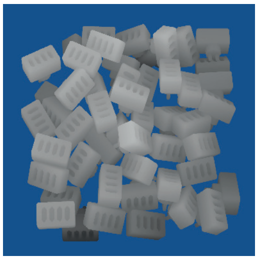
(a)

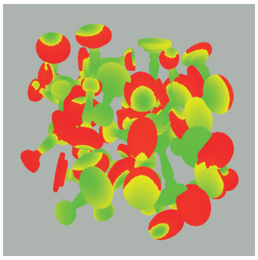
(b)

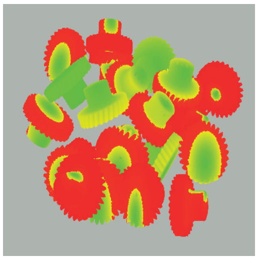
(c)

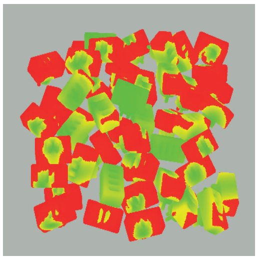

(d)

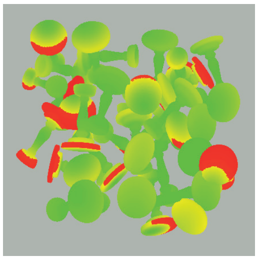

(e)

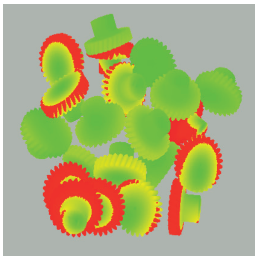

(f)


(g)


(h)


(i)


(j)


(k)

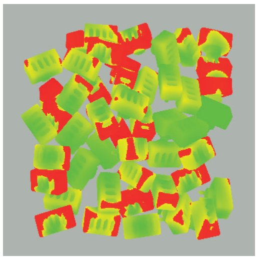

(l)

Figure 5.10: Visualization for pose estimation results on bin-picking scenarios: (a-c) 3D point cloud inputs; (d-f) ground truth poses; (g-i) retrieved pose hypotheses by our baseline method; (j-l) retrieved pose hypotheses by the proposed method. Color-coded visualization of point-wise distance error ranging from 0 (green) to greater than 0.2 times the diameter of the object (red). Predicted poses with ADD more than 0.2 times the diameter of the object have been removed for visualization purposes.

5.6 Conclusions

In this chapter, we introduced a new method for 6D object pose estimation from 3D point clouds. Our core idea is to employ correlation between poses of object instances and object parts to improve the performance of object pose estimation. We make use of the self-attention mechanism and feature fusion to model the part-to-part and object-to-object relationships, and propose two modules $\mathtt{M}_\mathtt{P}$ and $\mathtt{M}_\mathtt{O}$. We first produce a number of proposals using part-centric and object-centric voting-based VoteNet. We then allow higher-order feature interactions between proposals via the non-local network CGNL. Ablation studies demonstrate the effectiveness of the proposed modules to improve the accuracy of pose estimation. Experiments further show that our architecture outperforms previous approaches. In Chapters 4 and 5, we have introduced methods for 6D object pose estimation that align CAD models of objects to input point clouds. For robot grasping, after performing model-to-scene alignment, a set of grasp configurations can be selected from a database of pre-computed grasps. However, this model-based approach is not able to synthesize grasps for novel objects due to the unavailability of 3D object model. The next chapter introduces an end-to-end trainable grasp generation network given an input 3D point cloud. We will show that the proposed approach is able to generalize and perform well on novel objects.

Chapter 6
Grasp Generation in Cluttered Scenes

6.1 Introduction

To provide grasp configurations for robot manipulation, conventional model-based grasp planning requires knowledge of the 6D pose of objects in the scene. A set of grasps is then selected from a database of pre-computed grasps [9]. In the previous chapters, we have introduced methods for 6D object pose estimation that register a CAD model of the object to be grasped to measured data. However, predicting the pose of novel objects is not possible, as for 6D object pose estimation we assume that the 3D model of the object is available and crucially that the object coordinate system is defined in the 3D space of the model. Even if we can estimate the 6D pose of unknown objects, synthesizing grasps for these objects is unachievable, as for the model-based approaches we assume that the 3D model of objects is available.

An alternative approach is to generate the grasp configurations directly from sensor data without assuming a known 3D model of the object or pre-computed grasps [75]. Inspired by the success of convolutional neural networks (CNNs) in a broad range of computer vision tasks, recent works [75, 76, 101, 69] rely entirely or partially on deep learning. Some methods only employ deep CNNs for finding features of a good grasp from data [75, 76], while others employ end-to-end learning for grasp generation [101, 69]. The reported results from both are promising across a wide variety of objects, sensors, and robot end effectors. However, the current state-of-the-art CNN-based grasp generation methods utilize 2D or 2.5D input without taking the 3D geometry information into consideration. This might lead to failure to perform a grasp due to the lack of geometric analysis. Therefore, a few approaches have been proposed to localize grasps from 3D point sets [116, 84, 71, 29].

Although grasp generation methods in point clouds have achieved remarkable results, many problems remain unsolved. Due to measurement noise, occlusions, and undesirable contacts with the environment, generating feasible and reliable grasps in cluttered scenes is difficult. Many existing methods require time-consuming multi-stage processing for sampling grasp candidates and evaluating the grasp quality. While several works proposed end-to-end models for 6-DOF grasp generation and achieved state-of-the-art results in benchmarks, most of these methods rely only on features extracted by a backbone network such as PointNet++ [99] to predict grasps without considering the relationship between objects in the scenes. Grasping in clutter requires both reasoning about object parts and potential collisions with the gripper. Therefore, the contextual information, encapsulating the geometry of the rest of the scene, is important and should be taken into consideration to boost the performance of collision-free grasp generation in cluttered environments.

In this chapter, we propose an end-to-end deep learning approach for generating grasps on objects of interest, given point cloud measurements of the workspace. The core of our approach is to encode the positional relationship between objects in the scene into features by a context learning module. The contextual information enables our model to reduce the number of generated grasps that can cause the gripper to collide with other objects. To make the developed system robust to occlusion, we build our network on the ideas from Chapter 5. The voting mechanism allows our model to perform grasp generation under clutter and occlusion. The specific improvement in performance due to each of our contributions is quantitatively measured in both simulation and real-world evaluation.

The main contributions of this chapter can be summarized as follows: (1) A new framework for 6-DOF grasp generation named VoteGrasp, that robustly generates grasp configurations in cluttered environments under severe occlusion using a voting mechanism. (2) A context learning module encoding the dependency of objects in the scene into features to boost the performance of collision-free grasp generation. (3) Demonstration of the generalization capability of our method to novel objects.

6.2 Context and Attention in 3D Point clouds

Much prior work has widely explored the use of the contextual information to improve performance of 3D point matching [25], point cloud semantic segmentation [136], instance segmentation of 3D point clouds [55], and 3D scene layout prediction [107]. Deng et al. [25] introduced PPFNET that captures the global context across all local patches using a max-pooling aggregation and fusing the output into the local description to boost the local feature representation. Ye et al. [136] presents a pointwise pyramid pooling module to learn a multi-scale neighboring context. Hu et al. [55] achieve robust multiscale patch-based segmentation via exploiting the learned cluster-based contextual information. Shi et al. [107] show that using hierarchical context aggregation and propagation based on a denoising recursive autoencoder improves object detection performance on real-world 3D point cloud datasets.

Motivated by the success in natural language processing, recent works have focused on leveraging the self-attention mechanism with contextual dependency to achieve more accurate results in various perception tasks. Xie et al. [134] connects the self-attention idea with shape context to propose ShapeContextNet that can be applied to the general point cloud classification and segmentation problems. Zhang et al. [139] proposed a point contextual attention network for point cloud based retrieval. It takes the local point features and produces an attention map that enables the network to find more important features and produce a more discriminative global descriptor. Paigwar et al. [90] use a visual attention mechanism with point clouds to achieve accurate detection of objects. The core idea behind attention mechanisms is to pay more attention to the related parts of input with respect to the task. Grasping in clutter requires both reasoning about object parts and potential collisions with the gripper. Therefore, we find that self-attention based context learning is well suited to our problem of interest. Similar to Chapter 5, we opt for the compact generalized non-local network (CGNL) [137] as our self-attention module.

6.3 3D Point Cloud Based Grasp Generation

Machine learning-based approaches have been introduced to generate grasps from 3D point clouds [116, 84, 71, 88], with promising results across a wide variety of objects, sensors, and robots. ten Pas et al. [116] proposed a grasp generation method that first generates a large set of grasp hypotheses by a sampling process and then classifies them as good or bad grasps. The authors solve the binary classification task using a four-layer convolutional neural network (CNN). The CNN-based classifier is constructed by several projection features on a normalized point cloud. Extending on the idea of GPD, PointNetGPD [71] replaces the CNN-based grasp quality evaluation model by an evaluation network using the architecture of PointNet [98]. Although both methods [116, 71] densely sample candidates, they are not able to generate grasps on regions such as rims of mugs or plates where they can not estimate surface normals correctly. To overcome this limitation, [84] considers grasp generation as sampling a set of grasps using a variational autoencoder, then assesses and refines the sampled grasps using a grasp evaluator network. The variational autoencoder is trained to map the 3D point cloud of an observed object to a diverse set of grasps for the object. The evaluator model maps the point cloud and the robot gripper to a quality assessment of the 6D gripper pose using the PointNet [98] architecture. However, this approach only focuses on local features around the grasped object. To encode global information, PointNet++ Grasping [88] abandons the conventional learning pipeline and takes the whole scene point clouds as input to regress the grasp poses. However, only relying on features extracted by a backbone network such as PointNet++ [99], these methods lack the consider-

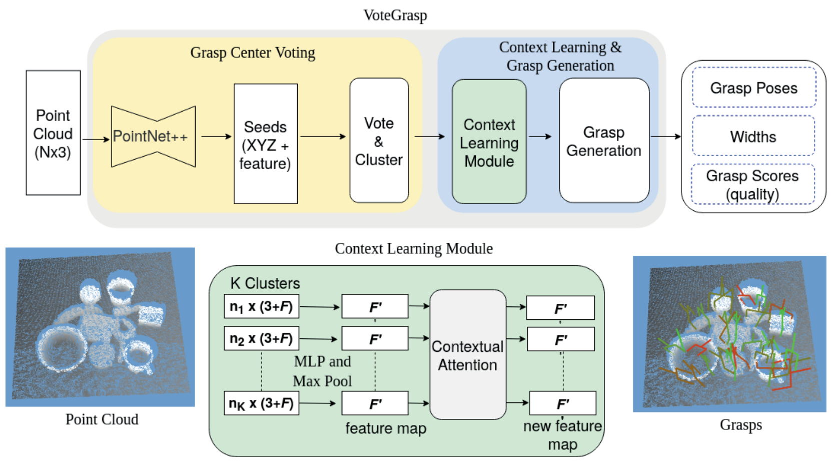


Figure 6.1: The architecture of the proposed VoteGrasp for 6-DOF grasp generation in point cloud data. Our model builds on a deep Hough voting neural network [97] to vote grasps with an added self-attention context learning module. Grasps are color-coded by the predicted quality scores. Green is the highest and red is the lowest.

ation of the relationships between different objects, which limits their performance in cluttered scenes. As a result, they have not yet been demonstrated to be reliable under occlusion, which is common in manipulation domains. We address this challenge by leveraging a voting mechanism and contextual information to generate grasp configurations directly from 3D point clouds.

6.4 VoteGrasp Approach

We introduce our end-to-end grasp generation network given scene point cloud inputs, which is illustrated in Fig. 6.1. In this work, we address the problem of generating grasps for any desired object in a cluttered scene from partial point cloud observations. The input to our approach is a point cloud of size $N \times 3$. The network aims to predict a ranked list of grasps, where each grasp $G = (p, R, w, q)$ specified by a center $p = (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}^3$, the gripper orientation $R \in SO(3)$, a width of the gripper $w \in \mathbb{R}$, and a grasp quality measure $q \in [0, 1]$. Due to the non-linearity of the rotation space as explained by [94], directly regressing the 3D orientation is difficult. Therefore, we reformulated the gripper orientation estimation as in [60]. We decouple orientation prediction into first recovering a viewpoint anchor (a discrete viewpoint classification task) and then estimating an in-plane rotation (Fig. 6.2) as a mixture of classi-

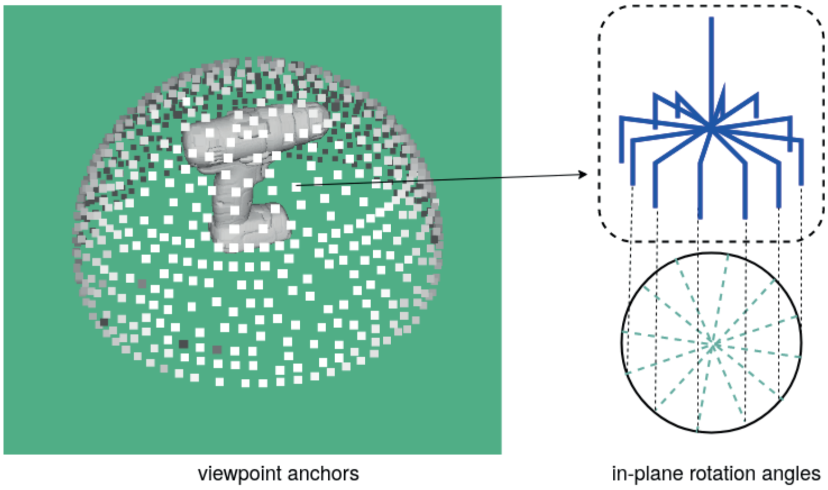


Figure 6.2: The orientation of the gripper is defined by a viewpoint anchor (a discrete viewpoint) and an in-plane rotation angle. The left part of the figure shows anchors sampled in the upper hemisphere. The right part visualizes in-plane rotation angles.

fication and regression formulations. In the rest of this section, we will examine each of the main components of our proposed architecture.

Backbone Network: In order to extract geometric features, we utilize the PointNet++ architecture with multi-scale grouping as our backbone network. Thereby, we are able to capture fine geometric structures from the neighborhood of each point. The backbone network selects M interest points (called seed points) and enriches them with high-dimensional features $\{s_i\}_{i=1}^{M}$ where $s_i = [x_i; f_i]$ with $x_i \in \mathbb{R}^3$ being the seed location in 3D space and $f_i \in \mathbb{R}^F$ being a feature vector.

Vote and Cluster: The seed points $\{s_i\}_{i=1}^{M}$ are then fed into a multi-layer perceptron (MLP) to compute votes $\{\{v_{ij} = [y_{ij}; g_{ij}] \in \mathbb{R}^{3+F}\}_{i=1}^{M}\}_{j=1}^{J}$, J votes per seed. The MLP consists of fully connected layers, ReLU and batch normalization. Each vote v_{ij} is represented by a point y_{ij} in 3D space with its Euclidean coordinates supervised to be close to a grasp center, and a feature vector g_{ij} learned for the final grasp generation task (F-dimensions). Our approach, VoteGrasp, computes multiple votes per seed $\mathbf{V} = \{v_j\}$ with $j = 1, .., J$. This is because we aim to estimate more than one grasp pose for each object. The next step is to cluster the votes by uniform sampling and finding neighboring votes within a certain Euclidean distance. Given input votes $\{v_i = [y_i; g_i] \in \mathbb{R}^{3+F}\}_{i=1}^{M \times J}$, we use iterative farthest point sampling (FPS) based on $\{y_i\}$ to choose a subset of K votes $\{v_{i_k}\}_{k=1}^{K}$. To find neighboring votes, a ball query finds all votes that are within a given radius to the query vote ν_{i_k}. The output are K clusters of vote sets of size $K \times n_k \times (3+F)$, where each group corresponds to a grasp center and n_k is the number of votes in the neighborhood of the vote ν_{i_k}.

Context Learning: Grasping in cluttered environments requires both reasoning about invisible object parts and potential collisions with the manipulator. Therefore, it is important to encode the relationship of objects in the scene into features or exploit contextual information outside of interest regions for generating collision-free grasps. However, the VoteNet architecture is designed to detect each object individually. Indeed each cluster $\mathcal{C}_k$ is independently pushed through the MLP layer to regress its object class and bounding box. Context outside a cluster is crucial and could help make more informed grasp predictions. Therefore, instead of processing each cluster independently to predict grasps, our network computes a new feature map from all clusters to learn the context that considers the relationships between all clusters. We find inspiration from self-attention based models [119, 134, 123, 31] to add a contextual module into our framework to capture the contextual information in 3D points. By leveraging a self-attention mechanism, we can combine features from other clusters to give more information on the object relationships. We opt for the compact generalized non-local network (CGNL) [137] as our self-attention module. The details of CGNL can be found in Chapter 5. CGNL allows for explicit modeling of rich interdependencies between clusters in feature space in a fast and low-complexity computation flow. More specifically, we first aggregate features from votes in each cluster. Votes $\{\nu_i = [y_i; g_i] \in \mathbb{R}^{3+F}\}_{i=1}^{n_k}$ in cluster k are fed into a MLP network before being max-pooled to a single feature vector $C_k \in \mathbb{R}^{F'}$. At this stage, we have a feature map $C = [\mathcal{C}_1; \mathcal{C}_2; ...; \mathcal{C}_K] \in \mathbb{R}^{K \times F'}$ from K clusters summarizing local context. In order to enable features to become aware of their global neighborhood, we explicitly model higher-order interactions between features in C, and it can be formulated as:

$$\mathcal{C}_{context} = \mathrm{CGNL}(\max_{i=1,...n}(\mathrm{MLP}(\nu_i))) \tag{6.1}$$

The self-attention mechanism allows the features from different clusters to interact with each other. The output is a new feature map of the same size $C_{context} = [\mathcal{C}_1^{ct}; \mathcal{C}_2^{ct}; ...; \mathcal{C}_K^{ct}] \in \mathbb{R}^{K \times F'}$. The effectiveness of the context learning module is visualized in Fig. 6.3. As we can see, when context is taken into account fewer of the grasps generated on a target object (banana) are in collision (shown in red) with neighboring objects.

Grasp Generation: Given a new feature map $C_{context} = [\mathcal{C}_1^{ct}; \mathcal{C}_2^{ct}; ...; \mathcal{C}_K^{ct}] \in \mathbb{R}^{K \times F'}$, a multi-layer perceptron network is applied to output a ranked list of grasps, where each grasp $G = (p, R, w, q)$ specified by a center $p = (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{R}^3$, the gripper orientation $R \in SO(3)$, a width of the gripper $w \in \mathbb{R}$, and a grasp quality measure $q \in [0, 1]$. To be specific, each $\mathcal{C}_k^{ct}$ is further processed

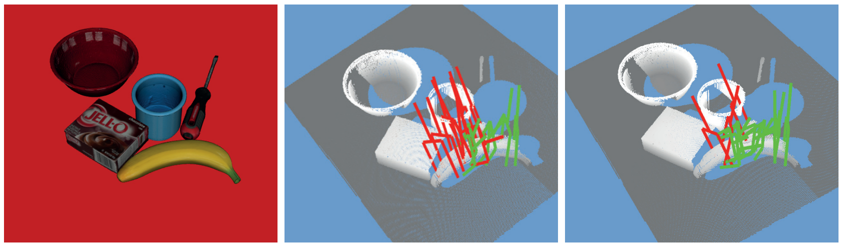

(a) (b) (c)

Figure 6.3: An example to illustrate the effectiveness of context learning module on grasp generation: (a) simulated scene; (b) result without the context module; (c) result with context module. The red grasps are not collision-free. Here we only visualize grasps for the target object (banana).

by a multi-layer perceptron composed of 3 fully connected layers. All fully connected layers are followed by batch normalization and ReLU except for the last prediction layer. The prediction layer has $5 + V + 2A$ channels where the output consists of 3 grasp center regression values, 1 gripper width regression value, 1 grasp confidence regression value, V viewpoint scores, A angle scores (in-plane rotation), and A angle residual regression values (in-plane rotation). V and A denote the numbers of sampled viewpoints and in-plane rotations respectively.

Loss function: We supervise the learning of modules jointly with a multi-tasks loss:

$$L_{votegrasp} = L_{vote} + L_{grasp} \tag{6.2}$$

The VoteGrasp loss $L_{votegrasp}$ includes a voting loss L_{vote} and a grasp estimation loss L_{grasp}. To supervise the learning of votes $\{v_i = [y_i; g_i] \in \mathbb{R}^{3+F}\}_{i=1}^{M \times J}$, we apply a regression loss:

$$L_{vote} = \frac{1}{M_o} \sum_i \|y_i - c_i^g\|_H \cdot \mathbb{1}(x_i) \tag{6.3}$$

where M_o is the total number of seeds on the object surface, c_i^g is the closest ground truth grasp center, $\| \cdot \|_H$ is the Huber norm and $\mathbb{1}(\cdot)$ is a binary function indicating whether a seed point s_i belongs to an object. We define the grasp loss function as follows:

$$L_{grasp} = L_{center} + \alpha L_{rot} + \beta L_{width} + \gamma L_{score} \tag{6.4}$$

where α, β and γ are weights that scale the losses to similar scales. The grasp loss is composed of a grasp center loss L_{center} (regression), a rotation loss $L_{rot} = L_{viewpoint} + L_{in-plane}$, a gripper width loss L_{width} (regression), and a grasp confidence score L_{score} (regression). The loss $L_{viewpoint}$ is for viewpoint classification. Meanwhile, for the in-plane rotation estimation, we use a mixture of classification and regression formulations $L_{in-plane} = 0.1L_{angle-cls} + L_{angle-reg}$. For all regression loss components of L_{grasp} we use the robust L1-smooth loss [102], while for classification the standard cross entropy loss is employed.

With the loss function in Eq. 6.2, training our network does not require object category labels. During inference, the trained model will generate grasp configurations without the information about object categories. This is because our goal is to make the trained network generalized to novel objects. However, our network is flexible and can be altered to predict categories together with poses, widths, and quality scores of object grasps. To this end, we need to add a semantic classification loss L_{sem} to L_{sem} $L_{votegrasp}$ as follows:

$$L_{votegrasp} = L_{vote} + L_{grasp} + L_{sem} \tag{6.5}$$

6.4.1 Implementation Details

Table 6.1: Layer parameters of the PointNet++ [99] based feature learning network.

layer name	input layer	layer params
SA1	point cloud	(2048,0.025,[64,64,128])
SA2	SA1	(1024,0.05,[128,128,256])
SA3	SA2	(512,0.1,[128,128,256])
SA4	SA3	(256,0.2,[128,128,256])
FP1	SA3, SA4	[256,256]
FP2	SA2, SA3	[256,256]

Network Architecture. In our implementation, we randomly choose N=50k points from each raw point cloud and set α=β=γ=1.0 in Eq. 6.4. We then apply the PointNet++ [99] based feature learning network, which has 4 set abstraction layers (SA) and 2 feature propagation layers (FP). The detailed layer parameters are shown in Table 6.1. The FP2 outputs $M = 1024$ seeds with $F = 256 - dim$ features and 3D coordinates that will be transformed to votes. The voting module generates $J = 10$ votes per seed with an MLP layer spec: $[256, 256, 259 \times 10]$. In the context module, we form $K = 1024$ clusters and output a new feature map $C_{context} \in K \times F'$ where $K = 1024, F' = 128$. In the last step, 1024 grasps are generated from the new feature map. The prediction layer has $5 + V + 2A$ channels where $V = 120$, and $A = 6$.

6.5 Evaluation

In this section, we first aim to determine to what extent the proposed network learns to synthesize grasps for target objects from a synthetic training set, and how well the trained model works in the real world. To achieve this, we introduced a pipeline to rapidly generate a large set of labeled data and trained the proposed network with the loss $L_{votegrasp}$ in Eq. 6.5. Then we evaluate the performance of the trained model on a synthetic test set and on real-world robotic grasping.

Furthermore, we are particularly interested to evaluate how well the learned model generalizes to novel object categories and in what way it compares to current state of the art. Finally, we evaluate the robustness of our approach to clutter and explore to what extent the use of context learning can mitigate the negative effects of occlusions.

To answer the above questions, we evaluate our method and compare with other state-of-the-art methods on the public dataset GraspNet-1Billion [29]. This is a large-scale grasp dataset collected from cluttered scenes considering multi-object-multi-grasp setting. The objects in GraspNet-1Billion have varying shapes, textures, sizes, materials and under different occlusion conditions. Hence, it can be used to evaluate robustness to occlusion and the generalization ability of our trained model (using the loss $L_{votegrasp}$ in Eq. 6.2).

6.5.1 Synthetic Data Generation

In order to train our VoteGrasp framework, a dataset is needed that contains point clouds as well as ground-truth grasps and votes. Since labeling data manually is error-prone and tedious, we develop an automated pipeline for synthetic point cloud generation and grasp pose annotation which runs on the platforms provided by Blender [20] and the GraspIt! [81] as shown in Fig. 6.4. The synthetic point cloud generation method samples training examples using two distributions. The first distribution is a state distribution that randomizes over object categories, object poses, and camera parameters. The second distribution is an observation distribution that models sensor operation and noise. We sample synthetic depth images using rendering and then compute point clouds from these images. In addition, object masks are also extracted by determining the set of pixels in the depth images with corresponding 3D points on the surface of objects.

Regarding grasp annotation, we use a gripper model of the Franka Emika Panda robot, but this method could be applied to other grippers as well. After robot gripper and target object models have been loaded into a virtual 3D workspace, we apply the eigengrasp concept [19] and collision detection system

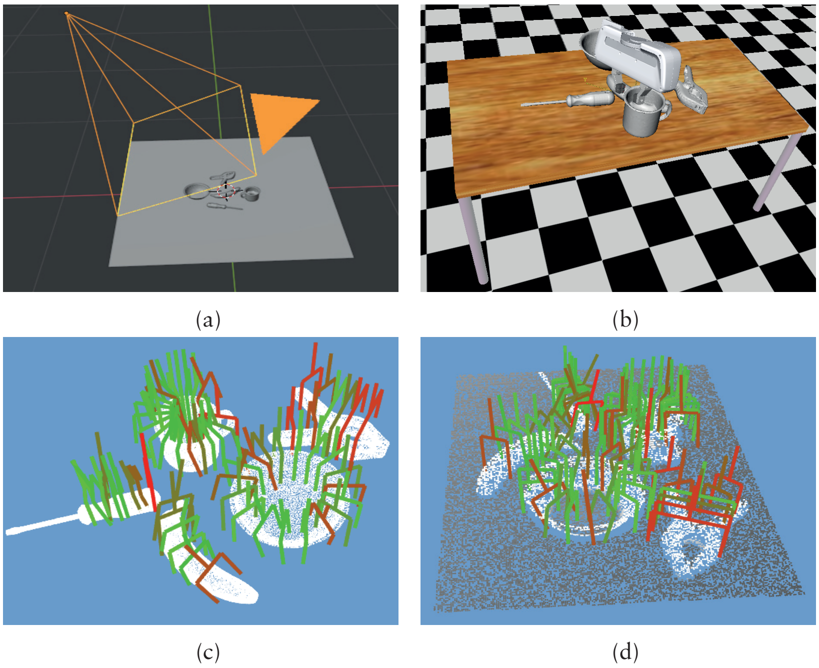


Figure 6.4: Training data generation: (a) synthetic point cloud generation; (b) grasp pose annotation; (c) object models and grasps; (d) synthetic point cloud and grasps. Grasps are color-coded by the quality scores. Green is the highest and red is the lowest.

in Graspit! [81] to produce stable grasps. Then we follow [71, 29] to compute a quantitative score of grasp quality.

This pipeline enabled us to collect a dataset of 100k synthetic scene point sets containing 5 million ground-truth grasps of 10 objects from the YCB benchmark set [12] in just a day. There are 90k training samples and 10k test samples, where the training samples are generally divided into the train split (75k samples) and the validation split (15k samples). The 10 objects are selected from the YCB dataset [12] which has been commonly used in robotic grasping research. We train the entire network (using the loss $L_{votegrasp}$ in Eq. 6.5) over 200 epochs with stochastic gradient descent using a batch size of 8 and the Adam optimizer with a learning rate of 0.001. It takes around 30 hours for training on one Nvidia GeForce RTX 2080 Ti 10GB GPU.

6.5.2 GraspNet-1Billion

The GraspNet-1Billion [29] consists of 97,280 RGB-D images captured from 190 cluttered scenes. The dataset provides over one billion grasp poses for 88 objects presented in the scenes. An accurate 3D mesh model of each object is available as well. Besides, they also provide camera poses, 6D object poses, object masks and bounding boxes for all frames. The rich annotations allow us to generate ground truth votes and grasp configurations easily. Following [29] we split the dataset into 100 scenes for training and 90 scenes for testing. To evaluate model generalizability, the test sets are divided into 30 scenes with novel objects, 30 for unseen but similar objects, and the rest for seen objects.

The proposed network is trained from scratch in an end-to-end manner. We train the entire network (using the loss $L_{votegrasp}$ in Eq. 6.2) over 200 epochs with stochastic gradient descent using a batch size of 8 and the Adam optimizer with a learning rate of 0.001. It takes around 80 hours for training on one Nvidia GeForce RTX 2080 Ti 10GB GPU. For inference, the forward-pass time of VoteGrasp for a single scene with size of 50k points is 150ms.

6.5.3 Evaluation of learning performance

We evaluate the learning performance on the synthetic test set. Average precision (AP) is used as our evaluation metric, which measures the precision of predicted grasps. We first check whether a grasp (G_p) is true positive or not. It is considered a true positive only if it satisties three conditions: (i) the absolute difference of the predicted and ground-truth quality scores $\Delta < 0.3$; (ii) distance between predicted grasp point and ground-truth grasp point $d < 0.03$; (iii) it has an IOU_g of at least σ with the ground-truth grasp. The IoU_g is defined as the area of the intersection divided by the area of the union of a ground-truth grasp and a predicted grasp (G_p) projected to (G_{gt}) surface:

$$IoU_g = \frac{area(projection(G_p)) \cap area(G_{gt})}{projection(G_p)) \cup area(G_{gt})} \tag{6.6}$$

$area(G)$ is the area of the region bounded by gripper as shown in Fig. 6.5.

Table 6.2 shows learning performance of the trained model (AP) for the ten objects from the YCB dataset as mentioned in Section 6.5.1. We observe that we can correctly recover 30.9% and 52.7%of the original ground-truth grasps under IOU_g with threshold of σ=0.5 and σ=0.25 respectively.

6.5.4 Evaluation on GraspNet-1Billion

We follow prior work [29] and evaluate our result on the dataset using $Precision@k$. This metric measures the precision of top-k ranked grasps. We first check whether a predicted grasp (G_p) is true positive or not. It is considered a true positive only

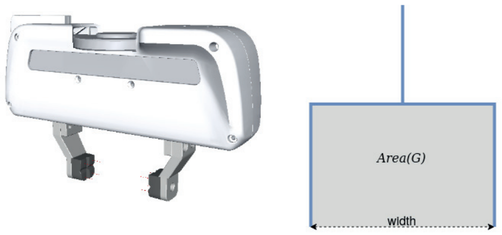


Figure 6.5: An example of grasp point and area of grasp $area(G)$.

Table 6.2: Quantitative evaluation (AP) of learning performance. The table shows the results on sythetic validation set. Note that the test set includes cluttered scenes.

	$AP_{IoU_g \geqslant 0.5}$	$AP_{IoU_g \geqslant 0.25}$
007_tuna_fish_can	30.2	53.1
008_pudding_box	32.7	54.9
011_banana	32.6	54.3
024_bowl	33.1	55.3
025_mug	28.0	48.7
044_flat_screwdriver	30.0	52.1
051_large_clamp	30.9	52.9
055_baseball	33.4	55.7
061_foam_brick	32.2	54.0
065-h_cups	25.6	46.1
MEAN	30.9	52.7

if the grasp satisfies three conditions: (i) there is an object inside the gripper; (ii) it is collision-free; (iii) the grasp is antipodal under a given friction coefficient μ. The third condition is computed based on the prior works [116, 29]. We let AP_μ denote the average $Precision@k$ for k ranges from 1 to 50 given a friction coefficient μ. We report the average of AP_μ with $\mu = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, denoted as **AP**.

Table 6.3 shows the performance of our approach compared to state of the art methods. We evaluated our trained model using the implementation of the evaluation metric shared by the authors of [29] enabling a direct comparison with the results of related works reported in [29, 37]. From the results presented in the table, we found that all the methods overall perform better in scenes with seen objects than scenes for novel objects. Notably, the AP score of our method surpasses the others in all the test sets by a large margin. Even on

Table 6.3: The table shows the results (**AP**) on GraspNet-1Billion test set captured by RealSense/Kinect sensors respectively. We let AP_μ denote the average $\mathtt{Precision@k}$ for k ranges from 1 to 50 given a friction coefficient μ. We report the average of AP_μ with $\mu = \{0.2, 0.4, 0.6, 0.8, 1.0\}$, denoted as **AP**. $\mathtt{Ours}^-$ denotes our proposed network without context learning module.

	Seen	Unseen (but similar)	Novel
GG-CNN [83]	15.5/16.9	13.3/15.1	5.5/7.4
Chu et al. [18]	16.0/17.6	15.4/17.4	7.6/8.0
GPD [116]	22.9/24.4	21.3/23.2	8.2/9.6
PointNetGPD [71]	26.0/27.6	22.7/24.4	9.2/10.7
GraspNet [29]	27.6/29.9	26.1/27.8	10.6/11.5
RGBD-Grasp [37]	28.0/32.1	27.2/30.4	12.3/13.1
$\mathtt{Ours}^-$	29.2/33.8	28.3/31.7	13.6/15.0
Ours	**34.1/37.5**	**33.0/35.9**	**16.9/18.5**

the scenes with novel objects, the proposed model still has an averaged 5.0% improvement over the best baseline [37]. This implies that our model is able to generalize and perform well on novel objects. Moreover, in order to evaluate the robustness of algorithms towards occlusion, we perform grasp generation under increasing levels of occlusion. To estimate the levels of occlusion, we calculate the visible surface ratio of each object instance. Fig. 6.6 illustrates how methods are influenced by different levels of occlusion. As shown, VoteGrasp performs well even when objects are heavily occluded, while the results of the previous approaches indicate high sensitivity to occlusion. Fig 6.7 shows example of test scenes in GraspNet-1Billion [29] dataset. Fig. 6.8 shows qualitative results of our predicted grasp poses.

6.5.5 Ablation Study

We validate the effectiveness of the self-attention contextual module of our network by comparing it with the model that directly generates grasps without context learning. According to results in Table 6.3, we see an improvement of 4.3%, from 31.5% to 35.8% for seen objects (averaged **AP** from both cameras). We also observe marked improvements, from 30.0% to 34.5% with unseen (but similar) objects and from 14.3% to 17.7% for novel objects. It confirms that our method greatly benefits from the use of contextual information. Furthermore, we evaluate the effects of the number of votes per seed J on the performance of our model. It seems that the model tends to perform better on seen objects with a smaller value of J, but does not generalize well for novel

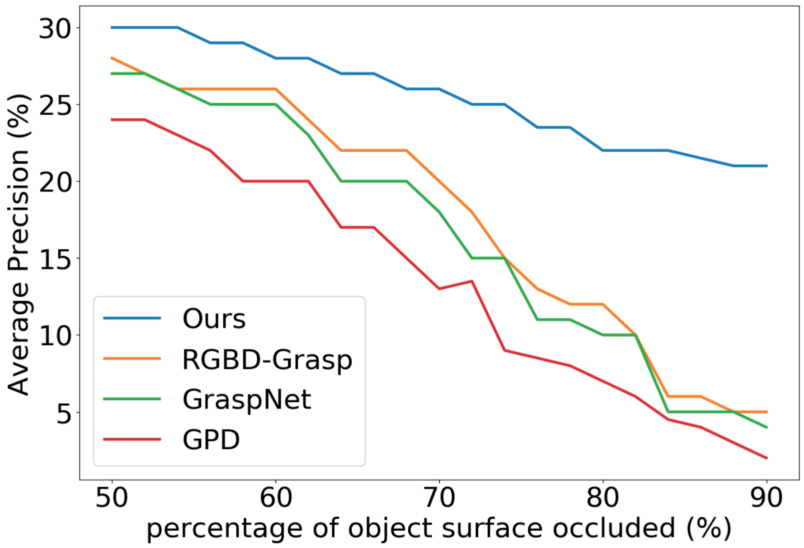


Figure 6.6: Performance of different approaches under increasing levels of occlusion. Our method performs well even when objects are heavily occluded, while the results of the previous approaches indicate high sensitivity to occlusion.

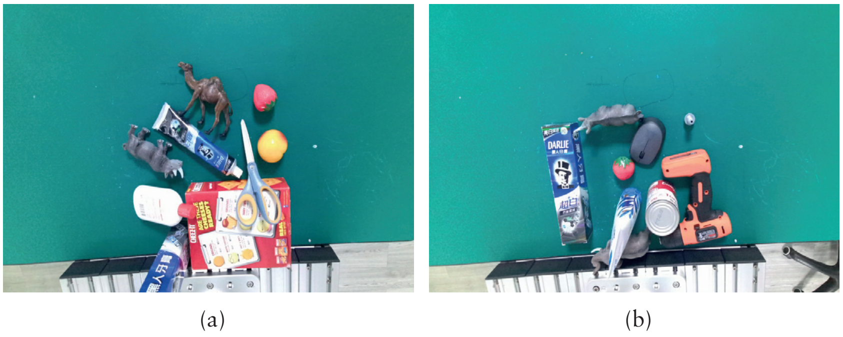
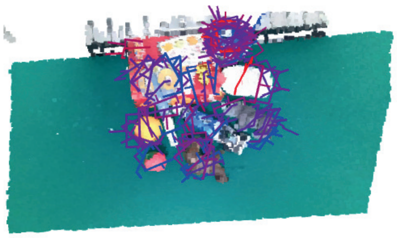


Figure 6.7: Example of test scenes in GraspNet-1Billion [29] dataset.

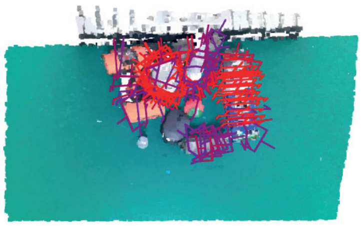
objects. We find that J = 10 votes per seed achieves the best results in scenes with novel objects, as shown in Table 6.4.

(a)

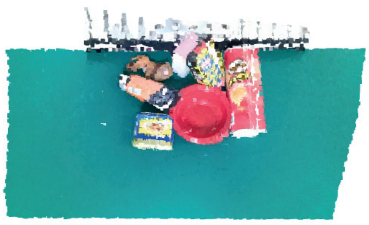


(b)

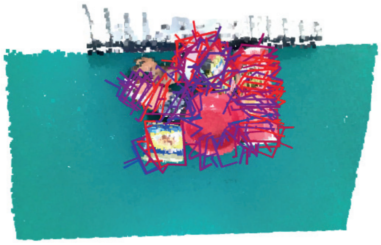
(c)



(d)



(e)



(f)

Figure 6.8: Examples of input point clouds and predicted grasps from our proposed method; (a-c-e) input point clouds in GraspNet-1Billion [29] dataset; (b-d-f) grasps generated by VoteGrasp. Grasps are color-coded by the confidence score. Red is the highest and blue is the lowest.

Table 6.4: Effects of number of votes per seed to the performance of our model. Evaluation metric is **AP** on GraspNet-1Billion [29].

	Seen	Unseen (but similar)	Novel
J=1	**34.8/38.0**	32.6/35.5	13.2/15.6
J=5	34.5/37.8	32.8/35.6	14.5/16.9
J=10	34.1/37.5	**33.0/35.9**	**16.9/18.5**
J=15	32.4/35.1	31.3/33.7	14.0/16.6
J=20	29.1/32.2	28.0/31.1	12.2/14.0

6.5.6 Robotic Grasping Experiment

The experiments were conducted with a Franka Emika Panda robot arm with 7-DOF, equipped with a parallel-jaw gripper as shown in Fig. 6.9. To capture the input point clouds, we used either ASUS Xtion PRO LIVE sensor or Microsoft Kinect sensor v2. The whole system is implemented using the ROS and MoveIt! frameworks.

In the first test, we evaluate our netwrotk trained only on synthetic data as described in Section 6.5.1. Thirty grasping rounds were run for each target object. The objects are randomly placed within the workspace of the robot arm and the camera. A grasp was considered a success if the robot could grasp and lift the object within one attempt. As shown in Table 6.5, our model was trained purely in simulation and works well in the real world without any extra steps.

Table 6.5: Results of real robot experiments. Our model was trained purely in simulation. The table shows number of attempts, number of successful attempts, and grasp success rate per target object.

	Attempt	Success	Success Rate
007_tuna_fish_can	30	21	70%
008_pudding_box	30	22	73%
011_banana	30	20	67%
024_bowl	30	21	70%
025_mug	30	22	73%
044_flat_screwdriver	30	17	57%
051_large_clamp	30	20	67%
055_baseball	30	19	63%
061_foam_brick	30	23	77%
065-h_cups	30	22	73%

Table 6.6: Results of real robot experiments. The networks were trained on the GraspNet-1Billion dataset. The table shows number of attempts, number of successful attempts, and grasp success rate. Methods: GPD [116], PointNetGPD [71], GraspNet [29], our proposed network without context learning module (Ours^-), and our method with full options (Ours).

Method	Attempt	Success	Success Rate
GPD [116]	150	100	67%
PointNetGPD [71]	150	102	68%
GraspNet [29]	150	106	71%
Ours^-	150	109	73%
VoteGrasp (Ours)	150	**117**	**78%**

In the next experiment, we evaluate the real-world grasping performance of our method, GPD [116], PointNetGPD [71], and GraspNet [29]. For a fair comparison, all models were trained with the GraspNet-1Billion dataset. Our VoteGrasp framework was trained as described in Section 6.5.2. For training GPD [116], PointNetGPD [71], and GraspNet [29], we set all hyperparameters as in the original papers. We selected 10 known YCB objects and 15 novel objects with shapes and sizes that fit the gripper. In each scene, we randomly select 10-15 objects and randomly place them on the table. One hundred and fifty grasping rounds were run for each method. We let the robot randomly pick objects. A grasp was considered a success if the robot could grasp and lift the object within one attempt. As shown in Table 6.6, our method outperforms previous state-of-the-art methods with remarkable margins. The results show that our method has a clear advantage of using the voting mechanism and the context learning module.

The execution times of the individual components, averaged over all real grasping rounds, are shown in Table 6.7. The numbers indicate that the proposed method is almost 19 times faster on average than the previous ones. GPD and PointNetGPD require sampling several thousand grasp candidates in the processing stage and classifying these candidates during inference. In the contrast, our approach directly generates grasps from votes without the grasp sampling process. Fig. 6.9 shows the grasping experiments carried on a Franka Emika Panda robot arm.

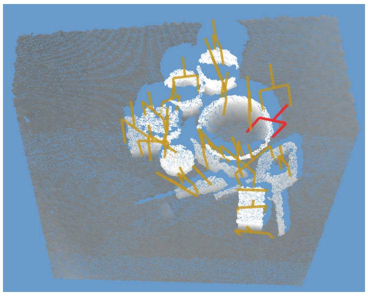
6.6 Conclusions

In this chapter, we introduced VoteGrasp, an end-to-end 6-DOF grasp generation network given 3D point clouds. The main contribution of this work is to show that by taking advantage of the deep Hough voting mechanism and

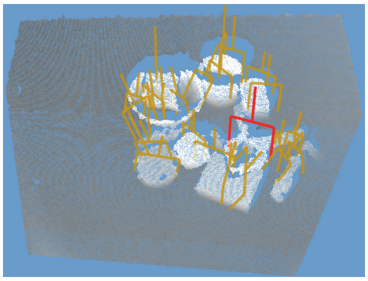
Table 6.7: Average run-time analysis of components. (ms per set of 50k points)

	Processing	Inference	Total
GPD	6.61	2.03	8.64
PointNetGPD	5.75	2.68	8.43
GraspNet [29]	0.30	0.65	0.95
Ours^{-}	0.30	**0.10**	**0.40**
VoteGrasp (Ours)	0.30	0.15	0.45

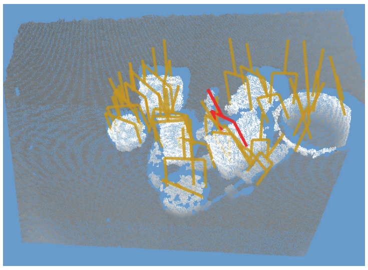
contextual information we are able to improve the performance of grasp generation compared with previous state-of-the-art methods. Through experiments, we demonstrate that VoteGrasp is highly robust to clutter and occlusions. Importantly, the results confirm that our proposed model is able to generalize and perform well on novel objects. Interesting future work is to consider adding a reachability predictor to the grasping network and discover the use of our approach in task planning applications.

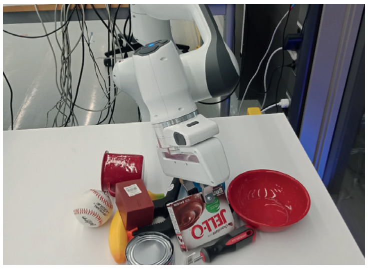

(a)

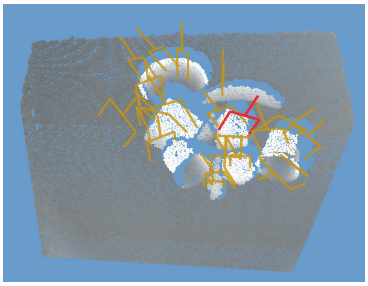
(b)


(c)

(d)


(e)


(f)


(g)

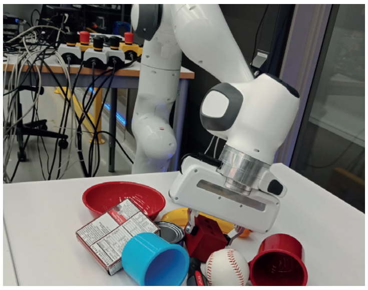

(h)

Figure 6.9: Real-world grasping experiment. (left) point clouds captured by an ASUS Xtion PRO LIVE sensor and grasps generated by the proposed model, red grasps are selected for execution; (right) Panda robot is grasping target objects.

Chapter 7
Conclusions

In this dissertation we have studied a variety of directions of vision-based perception for robot manipulation. Different components required for robot manipulation in unstructured environments were discussed including scene reconstruction, object pose estimation, and grasp generation. In this concluding chapter, the contributions are summarized. Then limitations of our approach are discussed followed by a presentation of the possible societal impact this work might have. Finally, some directions for future work are discussed.

7.1 Contributions

The principal contribution of this dissertation is the development of a number of vision-based perception methods specifically aimed towards autonomous robotic manipulation. In this subsection we summarize the four most important achievements of this work.

The first notable contribution of this work is the reliable camera tracking method, proposed in Chapter 3. A key message in this contribution is that by using only a single information channel no algorithm is completely robust to all practical conditions. Rather, we gain robustness by adaptively integrating cues from different channels. We demonstrated that by combining geometric, appearance, and semantic cues in the registration process we are able to obtain reliable camera tracking and state-of-the-art surface reconstruction.

A second important contribution of this dissertation is the proposed multi-view object pose estimation framework in Chapter 4. We presented an approach for recognizing objects present in a scene and estimating their full pose by means of an accurate 3D semantic reconstruction. While the main trend in CNN-based 6D pose estimation has been to infer object's position and orientation from single views of the scene, our approach explores performing pose estimation from multiple viewpoints, under the conjecture that combining multiple predictions can improve the robustness of the system. Rather than directly use raw depth and color frames like previous approaches, we employ the surfel-splatted predicted depth and color images of our accurate reconstruction scene model. Similarly, instead of operating on masks from segmentation, we use predicted 2D masks that are obtained by reprojecting of the current semantic map. As a result, our object pose estimation method benefits from the use of more accurate segmentation results as well as high-quality scene model. We demonstrated that we can exploit multiple viewpoints around the same object and the accurate 3D semantic reconstruction to achieve robust and stable 6D pose estimation in the presence of heavy clutter and occlusion.

The third contribution is an end-to-end approach for object pose estimation given a point cloud containing only geometric information. Our architecture pools geometric features together using a self-attention mechanism and adopts a deep Hough voting scheme for pose proposal generation. To build robustness to occlusion, the proposed network generates candidates by casting votes to accumulate evidence for object locations. Specifically, our model learns higher-level features in addition to the lower-level point features by encoding the dependency of object parts and object instances into features to boost the performance of object pose estimation. Experimental results show our method outperforms the state-of-the-art methods by large margins on public benchmark datasets.

Last but not least, this work proposed an end-to-end deep learning approach for generating 6-DOF collision-free grasps, based on 3D point cloud observations of the scene. The core of our approach is to encode the positional relationship between objects in the scene into features by a context learning module. The contextual information enables our model to increase the likelihood that the generated grasps are collision-free. To make the developed system robust to occlusion, we built our approach on top of the deep Hough voting architecture. Through experiments, we demonstrate that our method is highly robust to clutter and occlusions. Importantly, the results confirm that our proposed model is able to generalize and perform well on novel objects.

7.2 Societal and Ethical Impacts

This thesis develops algorithms that are relevant to vision-based perception for robot manipulation. The proposed methods are essential for enabling independent, safe and reliable operation in real-world unstructured environments. The developed technologies can be integrated into more complex systems for applications in various fields such as manufacturing, logistics, agriculture, or health care. These applications would create a large social impact. It is expected a reduction in operating costs, a consistently high-quality finish for every product produced, an improvement of working condition for employees, a reduced waste and increased yield, and a sustainable development.

Though we are still a few years away from having a personal healthcare companion like fictional the Baymax in the movie “Big Hero 6”, robots have been proving very useful in the medical sector. During the current pandemic, we have seen robots become essential workers in the COVID-19 response. They are deployed to deliver food and medicine to patients with COVID-19, sanitize hospitals, as well as help with social distancing. Besides positive impacts, there are also social and ethical challenges that arise with the development of robotic systems such as the future of the professions [113], unemployment [118], the use of robots in warfare [4]. Robot can not be danger by itself. It is our responsibility to use technology for the right purposes.

7.3 Limitations

The proposed algorithms in this dissertation have been verified through experimental validation on a number of public datasets and real robot experiments. Despite of promising results, it is important to note that there are also some limitations to the methods.

In Chapters 3 and 4, semantic information is a key element for improving camera pose tracking and mapping. Deep neural network models for image segmentation reliably producing pixel-wise semantic labels have intense computational requirements. This can be only satisfied with powerful GPUs, while mapping systems also require a high-powered GPU device to perform the processing of enormous amounts of data. This makes it difficult to integrate the proposed algorithm into applications with the use of an onboard perception system, autonomous mobile robots for example. For the work presented in this dissertation to be of practical applicability to those systems, GPU devices should be less expensive, smaller in size, and have lower power consumption. Such GPUs will most likely be available in the near future.

In Chapters 4 and 5, the developed object pose estimation methods are designed to deal with known rigid objects. They can not handle deformable objects which is important for application areas such as robotic surgery or agriculture. In addition, the proposed algorithms tend to focus on static environments, where the only motion is that of the camera. They might perform poorly in dynamic scenes where objects are moving independently.

In Chapter 6, grasp configurations generated by the proposed method can be used to pick up objects. However, manipulation is more than grasping. For instance, picking boxes from a pallet and loading them onto another pallet for delivery is common in logistics applications. Beyond grasp configurations, robots also need information about the boxes and the environment from the perception system to avoid undesirable contact with the environment and to optimize the positions of the boxes on the new pallet. Therefore, only grasp generation is not enough for those complex systems.

7.4 Future Research Directions

There are number of future directions we are interested in taking with the work in this dissertation. By looking at the limitations discussed above, one of the most significant issues with the current approach is how to deal with deformable objects. The object shape changes while forces are applied to the deformable body, which results in a variety of pose and appearance. Due to the large numbers state possibilities, estimating the pose of object or generating grasp configurations is very challenging. One possible approach would be to attempt to utilize multiple sensor modalities. While vision provdes global information about the shape, force and tactile sensors can cover both shape and contact in local areas. The vision-based perception algorithms proposed in this thesis can be a part of a more complex manipulation system for deformable objects.

Another problem not addressed in this thesis is how to reconstruct dynamic scenes where some object in the environment are moving. This might be possible if we extend the semantic mapping system in Chapter 4 to a dynamic object-level SLAM framework. While maintaining a global map of the static background, we can still track and fuse moving objects into separate maps. It is a multi-model SLAM system where each model is tracked and fused independently. By doing so hopefully we can model the shape of objects or track their motion over time.

Finally, in the context of multi-view object pose estimation with limited on-board resources, it may be interesting to investigate methods that can estimate camera poses without a powerful GPU. Rather than track the motion of camera by image registration, another possible approach would be to attempt to estimate camera viewpoints by matching individual object pose hypotheses across different views. We first obtain initial object candidates in single views, and then match these object candidates across views to build a single consistent scene. The resulting object-level correspondences could then be used to estimate the relative poses between the views. Another direction for future work on multi-view object pose estimation is to explore the next-best-view problem that maximize the object pose estimation accuracy over a given number of views. The Deep Hough voting approaches presented in Chapters 5 and 6 might be helpful in addressing next-best-view prediction.

References

[1] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *European conference on computer vision*, pages 511–524. Springer, 2012. (Cited on pages 55, 96, and 97.)

[2] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 585–592. IEEE, 2011. (Cited on pages 25, 55, and 87.)

[3] Vedrana Andersen, Henrik Aanæs, and Jakob Andreas Bærentzen. Surfel based geometry reconstruction. *TPCG*, 10:39–44, 2010. (Cited on page 12.)

[4] Ronald C Arkin. Ethical robots in warfare. *IEEE Technology and Society Magazine*, 28(1):30–33, 2009. (Cited on page 123.)

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*, 2015. (Cited on page 85.)

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. (Cited on page 18.)

[7] Tolga Birdal and Slobodan Ilic. Point pair features based object detection and pose estimation revisited. In *2015 International Conference on 3D Vision*, pages 527–535. IEEE, 2015. (Cited on pages 26, 27, and 87.)

[8] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):820–824, 1995. (Cited on page 34.)

[9] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis-a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013. (Cited on page 101.)

[10] Romain Brégier, Frédéric Devernay, Laetitia Leyrit, and James 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 Workshops*, pages 2209–2218, 2017. (Cited on pages xiv, xviii, 4, 73, 87, 93, 94, 95, 96, and 97.)

[11] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005. (Cited on page 85.)

[12] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017. (Cited on page 110.)

[13] Daniel R Canelhas, Todor Stoyanov, and Achim J Lilienthal. Sdf tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In *International Conference on Intelligent Robots and Systems*, pages 3671–3676. IEEE, 2013. (Cited on pages 11, 13, 31, 32, and 34.)

[14] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, and Ales Leonardis. G2l-net: Global to local network for real-time 6d pose estimation with embedding vector features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4233–4242, 2020. (Cited on pages 87, 96, and 97.)

[15] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. (Cited on pages 31, 32, and 34.)

[16] Changhyun Choi, Yuichi Taguchi, Oncel Tuzel, Ming-Yu Liu, and Srikumar Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *2012 IEEE International Conference on Robotics and Automation*, pages 1724–1731. IEEE, 2012. (Cited on pages 26 and 87.)

[17] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. (Cited on page 85.)

[18] Fu-Jen Chu, Ruinian Xu, and Patricio A Vela. Real-world multiobject, multigrasp detection. *IEEE Robotics and Automation Letters*, 3(4):3355–3362, 2018. (Cited on page 113.)

[19] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3270–3275. IEEE, 2007. (Cited on page 109.)

[20] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. (Cited on page 109.)

[21] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996. (Cited on page 11.)

[22] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. (Cited on page 10.)

[23] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017. (Cited on page 31.)

[24] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. (Cited on page 85.)

[25] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018. (Cited on page 102.)

[26] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern*

recognition, pages 998–1005. Ieee, 2010. (Cited on pages x, 25, 27, 55, 87, 96, and 97.)

[27] RO Duda and PE Hart. Use of the hough trans-form to detect lines andl curves in pictures. *Coin-mun. ACM*, 15. (Cited on page 83.)

[28] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *International Conference on Robotics and Automation (ICRA)*, pages 1691–1696. IEEE, 2012. (Cited on page 31.)

[29] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11444–11453, 2020. (Cited on pages xv, xix, 4, 101, 109, 110, 111, 112, 113, 114, 115, 116, 117, and 118.)

[30] Sergi Foix, Guillem Alenya, and Carme Torras. Lock-in time-of-flight (tof) cameras: A survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011. (Cited on page 9.)

[31] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. (Cited on pages 85 and 106.)

[32] Longsheng Fu, Fangfang Gao, Jingzhu Wu, Rui Li, Manoj Karkee, and Qin Zhang. Application of consumer rgb-d cameras for fruit detection and localization in field: A critical review. *Computers and Electronics in Agriculture*, 177:105687, 2020. (Cited on pages xvii and 10.)

[33] Ge Gao, Mikko Lauri, Yulong Wang, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. 6d object pose regression via supervised learning on point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3643–3649. IEEE, 2020. (Cited on pages 87, 96, and 97.)

[34] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. (Cited on page 18.)

[35] S Golemati, J Stoitsis, T Balkizas, and KS Nikita. Comparison of b-mode, m-mode and hough transform methods for measurement of arterial diastolic and systolic diameters. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 1758–1761. IEEE, 2006. (Cited on page 82.)

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. (Cited on pages ix, 14, and 18.)

[37] Minghao Gou, Hao-Shu Fang, Zhanda Zhu, Sheng Xu, Chenxi Wang, and Cewu Lu. Rgb matters: Learning 7-dof grasp poses on monocular rgbd images. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13459–13466, 2021. (Cited on pages 112 and 113.)

[38] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019. (Cited on pages xvii, 69, 70, and 72.)

[39] Jianwei Guo, Xuejun Xing, Weize Quan, Dong-Ming Yan, Qingyi Gu, Yang Liu, and Xiaopeng Zhang. Efficient center voting for object detection and 6d pose estimation in 3d point cloud. *IEEE Transactions on Image Processing*, 30:5072–5084, 2021. (Cited on pages 26 and 87.)

[40] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014. (Cited on pages 25, 55, and 87.)

[41] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, pages 1–1, 2020. (Cited on page 20.)

[42] Frederik Hagelskjær and Anders Glent Buch. Pointvotenet: Accurate object detection and 6 dof pose estimation in point clouds. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2641–2645. IEEE, 2020. (Cited on pages 87 and 97.)

[43] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian conference on computer vision*, pages 213–228. Springer, 2016. (Cited on page 37.)

[44] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. (Cited on pages 14, 61, 62, and 65.)

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (Cited on pages x, 14, and 19.)

[46] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020. (Cited on page 87.)

[47] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *International Conference on Robotics and Automation (ICRA)*, pages 2631–2638. IEEE, 2014. (Cited on pages 58 and 63.)

[48] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012. (Cited on pages 72, 94, 96, and 97.)

[49] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *European conference on computer vision*, pages 834–848. Springer, 2016. (Cited on pages 26 and 87.)

[50] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. (Cited on page 81.)

[51] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642, 1987. (Cited on page 46.)

[52] Paul VC Hough. Machine analysis of bubble chamber pictures. In *Proc. of the International Conference on High Energy Accelerators and Instrumentation, Sept. 1959*, pages 554–556, 1959. (Cited on page 83.)

[53] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654. (Cited on page 83.)

[54] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019. (Cited on page 85.)

[55] Shi-Min Hu, Jun-Xiong Cai, and Yu-Kun Lai. Semantic labeling and instance segmentation of 3d point clouds using patch context analysis and multiscale processing. *IEEE transactions on visualization and computer graphics*, 26(7):2485–2498, 2018. (Cited on page 102.)

[56] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016. (Cited on page 48.)

[57] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Robotics Research*, pages 235–252. Springer, 2017. (Cited on page 31.)

[58] Luca Iocchi, Domenico Mastrantuono, and Daniele Nardi. A probabilistic approach to hough localization. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 4250–4255. IEEE, 2001. (Cited on page 82.)

[59] H Kälviäinen. Motion detection using the randomised hough transform: exploiting gradient information and detecting multiple moving objects. *IEE Proceedings-Vision, Image and Signal Processing*, 143(6):361–369, 1996. (Cited on page 82.)

[60] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017. (Cited on page 104.)

[61] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European conference on computer vision*, pages 205–220. Springer, 2016. (Cited on page 83.)

[62] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013. (Cited on pages 12, 13, and 14.)

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on page 67.)

[64] K. Kleeberger, C. Landgraf, and M. F. Huber. Large-scale 6d object pose estimation dataset for industrial bin-picking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2573–2578, 2019. (Cited on pages xiv, xviii, 4, 93, 94, 96, and 97.)

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. (Cited on pages 17 and 19.)

[66] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. (Cited on pages xviii, 56, 73, 74, and 75.)

[67] Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén. Efficient multi-frequency phase unwrapping using kernel density estimation. In *European Conference on Computer Vision*, pages 170–185. Springer, 2016. (Cited on page 9.)

[68] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited on pages 17 and 19.)

[69] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. (Cited on page 101.)

[70] Deping Li, Hanyun Wang, Ning Liu, Xiaoming Wang, and Jin Xu. 3d object recognition and pose estimation from point cloud using stably observed point pair feature. *IEEE Access*, 8:44335–44345, 2020. (Cited on pages 26, 87, 96, and 97.)

[71] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3629–3635. IEEE, 2019. (Cited on pages xix, 101, 103, 110, 113, and 117.)

[72] Huei-Yung Lin, Shih-Cheng Liang, and Yu-Kai Chen. Robotic grasping with multi-view image acquisition and model-based pose estimation. *IEEE Sensors Journal*, 21(10):11870–11878, 2020. (Cited on page 56.)

[73] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. (Cited on page 67.)

[74] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. (Cited on page 62.)

[75] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017. (Cited on page 101.)

[76] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. (Cited on page 101.)

[77] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019. (Cited on page 81.)

[78] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. (Cited on pages xii, 42, 49, and 52.)

[79] John McCormac, Ronald Clark, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level slam. In *International Conference on 3D Vision (3DV)*, pages 32–41. IEEE, 2018. (Cited on pages 59 and 63.)

[80] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017. (Cited on pages xii and 58.)

[81] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. (Cited on pages 109 and 110.)

[82] Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in*

Neural Information Processing Systems, volume 27. Curran Associates, Inc., 2014. (Cited on page 85.)

[83] Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. In *Proc. of Robotics: Science and Systems (RSS)*, 2018. (Cited on page 113.)

[84] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2901–2910, 2019. (Cited on pages 101 and 103.)

[85] Fernando I Ireta Muñoz and Andrew I Comport. Point-to-hyperplane RGB-D pose estimation: Fusing photometric and geometric measurements. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 24–29. IEEE, 2016. (Cited on pages 32 and 34.)

[86] Yoshikatsu Nakajima and Hideo Saito. Efficient object-oriented semantic mapping with object detector. *IEEE Access*, 7:3206–3213, 2018. (Cited on pages xiii, 59, and 60.)

[87] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011. (Cited on pages 11, 12, 13, 31, 34, and 35.)

[88] P. Ni, W. Zhang, X. Zhu, and Q. Cao. Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3619–3625, 2020. (Cited on page 103.)

[89] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015. (Cited on page 59.)

[90] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1297–1306, 2019. (Cited on page 103.)

[91] Alessandro Palleschi, Marco Gugliotta, Chiara Gabellieri, Dinh-Cuong Hoang, Todor Stoyanov, Manolo Garabini, and Lucia Pallottino. Fully

autonomous picking with a dual-arm platform for intralogistics. In *Proceedings of the 2nd Italian Conference on Robotics and Intelligent Machines*, 2020. (Cited on pages xiii, 77, and 78.)

[92] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and P-P Sloan. Interactive ray tracing for isosurface rendering. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 233–238. IEEE, 1998. (Cited on page 12.)

[93] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2011–2018. IEEE, 2017. (Cited on page 27.)

[94] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570, 2019. (Cited on pages 27 and 104.)

[95] Hanspeter Pfister, Matthias Zwicker, Jeroen Van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342, 2000. (Cited on page 12.)

[96] RPK Poudel, S Liwicki, and R Cipolla. Fast-scnn: Fast semantic segmentation network. In *30th British Machine Vision Conference 2019, BMVC 2019*, 2019. (Cited on page 37.)

[97] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. (Cited on pages xiv, xv, xviii, 81, 83, 84, 88, 90, 95, and 104.)

[98] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. (Cited on pages x, 20, 22, 89, and 103.)

[99] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pages 5105–5114, 2017. (Cited on pages xviii, 20, 21, 93, 102, 103, and 108.)

[100] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017. (Cited on page 27.)

[101] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015. (Cited on page 101.)

[102] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. (Cited on page 108.)

[103] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018. (Cited on pages xvii, 59, 69, 70, and 72.)

[104] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. (Cited on pages 18 and 19.)

[105] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE, 2010. (Cited on pages 25, 55, and 87.)

[106] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. (Cited on pages 37 and 38.)

[107] Yifei Shi, Angel X Chang, Zhelun Wu, Manolis Savva, and Kai Xu. Hierarchy denoising recursive autoencoders for 3d scene layout prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1771–1780, 2019. (Cited on page 102.)

[108] Teresa M Silberberg, Larry Davis, and David Harwood. An iterative hough procedure for three-dimensional object recognition. *Pattern Recognition*, 17(6):621–629, 1984. (Cited on page 82.)

[109] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, (ICLR), San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. (Cited on pages 18 and 19.)

[110] Juil Sock, S Hamidreza Kasaei, Luis Seabra Lopes, and Tae-Kyun Kim. Multi-view 6d object pose estimation and camera motion planning using rgbd images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2228–2235, 2017. (Cited on page 56.)

[111] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 719–722. IEEE, 2011. (Cited on pages 13, 31, 32, and 36.)

[112] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012. (Cited on pages ix, xii, xvii, 9, 11, 42, 46, 47, 48, and 50.)

[113] Richard E Susskind and Daniel Susskind. *The future of the professions: How technology will transform the work of human experts*. Oxford University Press, USA, 2015. (Cited on page 123.)

[114] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. (Cited on pages 18 and 19.)

[115] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018. (Cited on pages 27 and 55.)

[116] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017. (Cited on pages xix, 101, 103, 112, 113, and 117.)

[117] Federico Tombari and Luigi Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, pages 349–355. IEEE, 2010. (Cited on pages 82 and 83.)

[118] Philippe Van Parijs. Basic income: a simple and powerful idea for the twenty-first century. *Politics & Society*, 32(1):7–39, 2004. (Cited on page 123.)

[119] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. (Cited on pages 85 and 106.)

[120] Hoang-Hiep Vu, Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE transactions on pattern analysis and machine intelligence*, 34(5):889–901, 2011. (Cited on page 40.)

[121] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019. (Cited on pages x, 14, 28, 30, 55, 66, 73, 75, and 87.)

[122] Haoyu Wang, Hesheng Wang, and Chungang Zhuang. 6d pose estimation from point cloud using an improved point pair features method. In *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, pages 280–284. IEEE, 2021. (Cited on pages 26 and 87.)

[123] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. (Cited on pages 85 and 106.)

[124] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. (Cited on page 20.)

[125] Oliver Wasenmüller and Didier Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision*, pages 34–45. Springer, 2016. (Cited on page 9.)

[126] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. In-hand scanning with online loop closure. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1630–1637. IEEE, 2009. (Cited on page 12.)

[127] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *International Conference on Robotics and Automation*

(ICRA), pages 5724–5731. IEEE, 2013. (Cited on pages 31, 32, 36, and 37.)

[128] Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous: Spatially extended kinectfusion. In *Proceedings of RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012. (Cited on pages 11, 12, 13, and 34.)

[129] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. (Cited on pages ix, 8, 12, 13, 31, 32, 37, 57, 58, 59, and 63.)

[130] Thomas J Whelan. *Real-time Dense Simultaneous Localisation and Mapping over Large Scale Environments*. PhD thesis, National University of Ireland Maynooth, 2014. (Cited on pages 13 and 37.)

[131] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE international conference on robotics and biomimetics*, pages 2987–2992. IEEE, 2011. (Cited on pages 25, 55, and 87.)

[132] Jay M Wong, Syler Wagner, Connor Lawson, Vincent Kee, Mitchell Hebert, Justin Rooney, Gian-Luca Mariottini, Rebecca Russell, Abraham Schneider, Rahul Chipalkatty, et al. Segicp-dsr: Dense semantic scene reconstruction and registration. *The Computing Research Repository (CoRR)*, abs/1711.02216, 2017. (Cited on pages 58 and 59.)

[133] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018. (Cited on pages 4, 27, 55, 67, and 73.)

[134] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018. (Cited on pages 103 and 106.)

[135] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 21–29, 2016. (Cited on page 85.)

[136] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 403–417, 2018. (Cited on page 102.)

[137] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *Advances in Neural Information Processing Systems*, pages 6510–6519, 2018. (Cited on pages xiv, 86, 89, 90, 91, 103, and 106.)

[138] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE, 2017. (Cited on pages xviii, 73, 74, and 75.)

[139] Wenxiao Zhang and Chunxia Xiao. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019. (Cited on page 103.)

[140] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018. (Cited on page 38.)

[141] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. (Cited on page 38.)

[142] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. (Cited on page 92.)

[143] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. (Cited on page 20.)

[144] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. (Cited on page 14.)

17. Berntsson, Tomas (2005) *Replacement of Lead Baths with Environment Friendly Alternative Heat Treatment Processes in Steel Wire Production.* Licentiate Thesis.

18. Tolt, Gustav (2005) *Fuzzy Similarity-based Image Processing.* Doctoral Dissertation.

19. Munkevik, Per (2005) *"Artificial sensory evaluation – appearance-based analysis of ready meals".* Licentiate Thesis.

20. Buschka, Pär (2005) *An Investigation of Hybrid Maps for Mobile Robots.* Doctoral Dissertation.

21. Loutfi, Amy (2006) *Odour Recognition using Electronic Noses in Robotic and Intelligent Systems.* Doctoral Dissertation.

22. Gillström, Peter (2006) *Alternatives to Pickling; Preparation of Carbon and Low Alloyed Steel Wire Rod.* Doctoral Dissertation.

23. Li, Jun (2006) *Learning Reactive Behaviors with Constructive Neural Networks in Mobile Robotics.* Doctoral Dissertation.

24. Otterskog, Magnus (2006) *Propagation Environment Modeling Using Scattered Field Chamber.* Doctoral Dissertation.

25. Lindquist, Malin (2007) *Electronic Tongue for Water Quality Assessment.* Doctoral Dissertation.

26. Cielniak, Grzegorz (2007) *People Tracking by Mobile Robots using Thermal and Colour Vision.* Doctoral Dissertation.

27. Boustedt, Katarina (2007) *Flip Chip for High Frequency Applications – Materials Aspects.* Doctoral Dissertation.

28. Soron, Mikael (2007) *Robot System for Flexible 3D Friction Stir Welding.* Doctoral Dissertation.

29. Larsson, Sören (2008) *An industrial robot as carrier of a laser profile scanner: Motion control, data capturing and path planning.* Doctoral Dissertation.

30. Persson, Martin (2008) *Semantic Mapping Using Virtual Sensors and Fusion of Aerial Images with Sensor Data from a Ground Vehicle.* Doctoral Dissertation.

31. Andreasson, Henrik (2008) *Local Visual Feature based Localisation and Mapping by Mobile Robots.* Doctoral Dissertation.

32. Bouguerra, Abdelbaki (2008) *Robust Execution of Robot Task-Plans: A Knowledge-based Approach.* Doctoral Dissertation.

33. Lundh, Robert (2009) *Robots that Help Each Other: Self-Configuration of Distributed Robot Systems.* Doctoral Dissertation.

34. Skoglund, Alexander (2009) *Programming by Demonstration of Robot Manipulators*. Doctoral Dissertation.

35. Ranjbar, Parivash (2009) *Sensing the Environment: Development of Monitoring Aids for Persons with Profound Deafness or Deafblindness*. Doctoral Dissertation.

36. Magnusson, Martin (2009) *The Three-Dimensional Normal-Distributions Transform – an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* Doctoral Dissertation.

37. Rahayem, Mohamed (2010) *Segmentation and fitting for Geometric Reverse Engineering. Processing data captured by a laser profile scanner mounted on an industrial robot.* Doctoral Dissertation.

38. Karlsson, Alexander (2010) *Evaluating Credal Set Theory as a Belief Framework in High-Level Information Fusion for Automated Decision-Making*. Doctoral Dissertation.

39. LeBlanc, Kevin (2010) *Cooperative Anchoring – Sharing Information About Objects in Multi-Robot Systems*. Doctoral Dissertation.

40. Johansson, Fredrik (2010) *Evaluating the Performance of TEWA Systems*. Doctoral Dissertation.

41. Trincavelli, Marco (2010) *Gas Discrimination for Mobile Robots.* Doctoral Dissertation.

42. Cirillo, Marcello (2010) *Planning in Inhabited Environments: Human-Aware Task Planning and Activity Recognition.* Doctoral Dissertation.

43. Nilsson, Maria (2010) *Capturing Semi-Automated Decision Making: The Methodology of CASADEMA*. Doctoral Dissertation.

44. Dahlbom, Anders (2011) *Petri nets for Situation Recognition*. Doctoral Dissertation.

45. Ahmed, Muhammad Rehan (2011) *Compliance Control of Robot Manipulator for Safe Physical Human Robot Interaction.* Doctoral Dissertation.

46. Riveiro, Maria (2011) *Visual Analytics for Maritime Anomaly Detection*. Doctoral Dissertation.

47. Rashid, Md. Jayedur (2011) *Extending a Networked Robot System to Include Humans, Tiny Devices, and Everyday Objects*. Doctoral Dissertation.

48. Zain-ul-Abdin (2011) *Programming of Coarse-Grained Reconfigurable Architectures*. Doctoral Dissertation.

49. Wang, Yan (2011) *A Domain-Specific Language for Protocol Stack Implementation in Embedded Systems*. Doctoral Dissertation.

50. Brax, Christoffer (2011) *Anomaly Detection in the Surveillance Domain*. Doctoral Dissertation.

51. Larsson, Johan (2011) *Unmanned Operation of Load-Haul-Dump Vehicles in Mining Environments*. Doctoral Dissertation.

52. Lidström, Kristoffer (2012) *Situation-Aware Vehicles: Supporting the Next Generation of Cooperative Traffic Systems*. Doctoral Dissertation.

53. Johansson, Daniel (2012) *Convergence in Mixed Reality-Virtuality Environments. Facilitating Natural User Behavior*. Doctoral Dissertation.

54. Stoyanov, Todor Dimitrov (2012) *Reliable Autonomous Navigation in Semi-Structured Environments using the Three-Dimensional Normal Distributions Transform (3D-NDT)*. Doctoral Dissertation.

55. Daoutis, Marios (2013) *Knowledge Based Perceptual Anchoring: Grounding percepts to concepts in cognitive robots*. Doctoral Dissertation.

56. Kristoffersson, Annica (2013) *Measuring the Quality of Interaction in Mobile Robotic Telepresence Systems using Presence, Spatial Formations and Sociometry*. Doctoral Dissertation.

57. Memedi, Mevludin (2014) *Mobile systems for monitoring Parkinson's disease*. Doctoral Dissertation.

58. König, Rikard (2014) *Enhancing Genetic Programming for Predictive Modeling*. Doctoral Dissertation.

59. Erlandsson, Tina (2014) *A Combat Survivability Model for Evaluating Air Mission Routes in Future Decision Support Systems*. Doctoral Dissertation.

60. Helldin, Tove (2014) *Transparency for Future Semi-Automated Systems. Effects of transparency on operator performance, workload and trust*. Doctoral Dissertation.

61. Krug, Robert (2014) *Optimization-based Robot Grasp Synthesis and Motion Control*. Doctoral Dissertation.

62. Reggente, Matteo (2014) *Statistical Gas Distribution Modelling for Mobile Robot Applications*. Doctoral Dissertation.

63. Längkvist, Martin (2014) *Modeling Time-Series with Deep Networks*. Doctoral Dissertation.

64. Hernández Bennetts, Víctor Manuel (2015) *Mobile Robots with In-Situ and Remote Sensors for Real World Gas Distribution Modelling*. Doctoral Dissertation.

65. Alirezaie, Marjan (2015) *Bridging the Semantic Gap between Sensor Data and Ontological Knowledge*. Doctoral Dissertation.

66. Pashami, Sepideh (2015) *Change Detection in Metal Oxide Gas Sensor Signals for Open Sampling Systems*. Doctoral Dissertation.

67. Lagriffoul, Fabien (2016) *Combining Task and Motion Planning*. Doctoral Dissertation.

68. Mosberger, Rafael (2016) *Vision-based Human Detection from Mobile Machinery in Industrial Environments*. Doctoral Dissertation.

69. Mansouri, Masoumeh (2016) *A Constraint-Based Approach for Hybrid Reasoning in Robotics*. Doctoral Dissertation.

70. Albitar, Houssam (2016) *Enabling a Robot for Underwater Surface Cleaning*. Doctoral Dissertation.

71. Mojtahedzadeh, Rasoul (2016) *Safe Robotic Manipulation to Extract Objects from Piles: From 3D Perception to Object Selection*. Doctoral Dissertation.

72. Köckemann, Uwe (2016) *Constraint-based Methods for Human-aware Planning*. Doctoral Dissertation.

73. Jansson, Anton (2016) *Only a Shadow. Industrial Computed Tomography Investigation, and Method Development, Concerning Complex Material Systems*. Licentiate Thesis.

74. Sebastian Hällgren (2017) *Some aspects on designing for metal Powder Bed Fusion*. Licentiate Thesis.

75. Junges, Robert (2017) *A Learning-driven Approach for Behavior Modeling in Agent-based Simulation*. Doctoral Dissertation.

76. Ricão Canelhas, Daniel (2017) *Truncated Signed Distance Fields Applied To Robotics*. Doctoral Dissertation.

77. Asadi, Sahar (2017) *Towards Dense Air Quality Monitoring: Time-Dependent Statistical Gas Distribution Modelling and Sensor Planning*. Doctoral Dissertation.

78. Banaee, Hadi (2018) *From Numerical Sensor Data to Semantic Representations: A Data-driven Approach for Generating Linguistic Descriptions*. Doctoral Dissertation.

79. Khaliq, Ali Abdul (2018) *From Ants to Service Robots: an Exploration in Stigmergy-Based Navigation Algorithms*. Doctoral Dissertation.

80. Kucner, Tomasz Piotr (2018) *Probabilistic Mapping of Spatial Motion Patterns for Mobile Robots*. Doctoral Dissertation.

81. Dandan, Kinan (2019) *Enabling Surface Cleaning Robot for Large Food Silo*. Doctoral Dissertation.

82. El Amine, Karim (2019) *Approaches to increased efficiency in cold drawing of steel wires*. Licentiate Thesis.

83. Persson, Andreas (2019) *Studies in Semantic Modeling of Real-World Objects using Perceptual Anchoring*. Doctoral Dissertation.

84. Jansson, Anton (2019) *More Than a Shadow. Computed Tomography Method Development and Applications Concerning Complex Material Systems*. Doctoral Dissertation.

85. Zekavat, Amir Reza (2019) *Application of X-ray Computed Tomography for Assessment of Additively Manufactured Products*. Doctoral Dissertation.

86. Mielle, Malcolm (2019) *Helping robots help us—Using prior information for localization, navigation, and human-robot interaction*. Doctoral Dissertation.

87. Grosinger, Jasmin (2019) *On Making Robots Proactive*. Doctoral Dissertation.

88. Arain, Muhammad Asif (2020) *Efficient Remote Gas Inspection with an Autonomous Mobile Robot*. Doctoral Dissertation.

89. Wiedemann, Thomas (2020) *Domain Knowledge Assisted Robotic Exploration and Source Localization*. Doctoral Dissertation.

90. Giaretta, Alberto (2021) *Securing the Internet of Things with Security-by-Contract*. Doctoral Dissertation.

91. Rudenko, Andrey (2021) *Context-aware Human Motion Prediction for Robots in Complex Dynamic Environments*. Doctoral Dissertation.

92. Eriksson, Daniel (2021) *Getting to grips with cartons: Interactions of cartonboard packages with an artificial finger.* Doctoral Dissertation.

93. Dinh-Cuong, Hoang (2021) *Vision-based Perception For Autonomous Robotic Manipulation*. Doctoral Dissertation.