University of Sheffield

An Intelligent Robotic Vision System with Environment Perception

Yixiang Jin

Supervisor: Dr Anthony Rossiter and Prof Sandor Veres

A report submitted in partial fulfilment of the requirements for the degree of *Doctor of Philosophy* *in the* Department of Automatic Control and Systems Engineering

August 7, 2022

Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: Yixiang Jin

Signature: Yixiang Jin

Date: 07/08/2022

Abstract

Ever since the dawn of computer vision[1, 2], 3D environment reconstruction and object 6D pose estimation have been a core problem. This thesis attempts to develop a novel 3D intelligent robotic vision system integrating environment reconstruction and object detection techniques to solve practical problems. Chapter 2 reviews current state-of-the art of 3D vision techniques from environment reconstruction and 6D pose estimation.In Chapter 3 a novel environment reconstruction system is proposed by using coloured point clouds. The evaluation experiment indicates that the proposed algorithm 2 is effective for small-scale and large-scale and textureless scenes. Chapter 4 presents Image-6D (that is section 4.2), a learning-based object pose estimation algorithm from a single RGB image. Contour-alignment is introduced as an efficient algorithm for pose refinement in an RGB image. This new method is evaluated on two widely used benchmark image data bases, LINEMOD and Occlusion-LINEMOD. Experiments show that the proposed method surpasses other state-of-the-art RGB-based prediction approaches. Chapter 5 describes Point-6D (defined in section 5.2), a novel 6D pose estimation method using coloured point clouds as input. The performance of this new method is demonstrated on LineMOD [3] and YCB-Video [4] dataset. Chapter 6 summarizes contributions and discusses potential future research directions. In addition, we presents an intelligent 3D robotic vision system deployed in a simulated/laboratory nuclear waste disposal scenario in Appendices B. To verify the results, a simulated nuclear waste handling experiment has been successfully completed via the proposed robotic system.

Acknowledge

First and foremost I am sincere grateful to my supervisor, Dr. Anthony Rossiter and Prof. Sandor Veres for their invaluable guidance, continuous support, and patience as I pursued my PhD. I would also like to thank Dr. Ayan Ghosh, Dr. James Clarke and Dr. Daniel Alonso for their technical support on my study. And for my girlfriend Cassie, thanks for all your love and support. Last but not the least, I want to thank my parents, Xiangming Jin and Hongyun Zhang, for everything they've done for me. Without their tremendous understanding and encouragement, it is impossible for me to finish my research.

List of Publication

Jin, Y. ; Rossiter, J. and Veres, S. (2021). *Accurate 6D Object Pose Estimation and Refinement in Cluttered Scenes*[1]. In Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems(ROBOVIS 2021), ISBN 978-989-758-537-1, pages 31-39. DOI: 10.5220/0010654500003061

Jin, Y.; A., Soto, D.A.P. ; Rossiter, J. and Veres, S. (2021). *Advanced Environment Modelling forRemote Teleoperation to Improve Operator Experience.* In Proceedings of the International Conference on Artificial Intelligence (icARTi '21), ISBN 978-1-4503-8575-6/21/12, DOI: 10.1145/3487923.3487939

[1]This paper received the best paper award at ROBOVIS'21.

Contents

List of Figures

List of Tables

List of Abbreviation

2D	2 dimension
3D	3 dimension
6D	6 degree of freedom
APE	absolute pose error
ADD	average distance
ADD-S	average distance for symmetric objects
AR	augmented reality
AUC	area under the ADD curve
BA	bundle adjustment
CA	contour alignment
CAD	computer-aided design
CNN	convolutional neural network
DOPE	deep object pose estimation
FPFH	fast point feature histograms
GUI	graphical user interface
ICP	iterative closet point
IMU	inertial measurement units
MLP	multi-layer perceptron

LM	levenberg-marquardt
LSTM	long short term memory
NDDS	NVIDIA Deep learning Dataset Synthesizer
ORB	oriented fast and rotated brief
PFH	point feature histograms
PnP	perspective-n-point
PTAM	parallel tracking and mapping
PVNet	pixel-wise voting network
RANSAC	random sample consensus
ROS	robot operating system
RCNN	region based convolutional neural networks
RPN	region proposal network
RPE	relative pose error
SFM	structure from motion
SHOT	signature of histograms of orientations
SIFT	scale-invariant feature transform
SLAM	simultaneous localization and mapping
STD	Standard Error
SURF	speeded up robust features
TSDFs	truncated signed distance functions
TOF	time of flight
UAV	unmanned autonomous vehicles
UR5	universal robot 5

List of Notations

$\mathbf{C}()$	colour intensity function
$\boldsymbol{d}_p()$	colour gradient of p
$\mathbf{f}$	camera focus length
$\mathbf{F}()$	objective function
$\boldsymbol{F}_G()$	geometric objective function
$\boldsymbol{F}_C()$	photometric objective function
$\boldsymbol{F}_x$	objective function
$\boldsymbol{F}_s$	source point cloud feature
$\boldsymbol{F}_t$	target point cloud feature
$\boldsymbol{J}_r$	jacobian matrix of r
$\mathbf{K}$	camera intrinsic matrix
$\mathbf{L}$	loss function
$\mathbf{L}()$	least-squares fitting
$\boldsymbol{n}_p$	normal of p
$\mathbf{p}$	point of source point cloud
$\bar{\boldsymbol{p}}$	nearest neighborhood of p
$\boldsymbol{P}_i$	ith point cloud
$\mathbf{q}$	point of target point cloud

$\boldsymbol{q}^{'}$	projection of q onto the tangent of p
$\mathbf{Q}$	target point cloud
$\mathbf{r}$	residual
$\mathbf{R}$	3D rotation
$\bar{\boldsymbol{R}}$	the ground truth of rotation
$\mathbf{T}$	3D translation
$\bar{\boldsymbol{t}}$	the ground truth of translation
$\boldsymbol{t}_x$	translation in x axis
$\boldsymbol{t}_y$	translation in y axis
$\boldsymbol{t}_z$	translation in z axis
$\mathbf{T}$	transformation matrix
$\boldsymbol{\phi}$	correspondence set
$\boldsymbol{\sigma}$	weight between F_G and F_C
$\boldsymbol{\mu}$	damping parameter
$\boldsymbol{\varphi}$	gain ratio
$\boldsymbol{\xi}$	pose increment

Chapter 1

Introduction

1.1 Background

The advanced robot is frequently depicted in science fiction films, which raises human's expectations of future robots. We want robots to perceive the surrounding world as we do. However, real-world machines cannot achieve such kinds of intelligence. Our human visual system can infer properties about 3D objects and understand the composition of surrounding environment quickly and effortlessly. In contrast, robot vision sensors capture an image of the environment, but an input device does not have perceptual capabilities or cognitive abilities; it does not know about the world. Therefore, to endow machines with these capabilities is, by far, a challenging task.

One of the most important goals of computer vision systems is to estimate 6D pose (the 3D translation and rotation) of object from complex environments. Accurate 6D pose estimation technologies can drive various emerging technological areas related to robotic manipulation, autopilot, augmented reality, etc. To perceive an accurate 6D pose in a cluttered scene, an amount of research has been dedicated to establishing an efficient 3D object perception system. Ideas range from feature-based methods [3, 8] which leverage object geometric information and learning-based approaches [13, 14, 15] which employ a deep neural network. No matter which paradigms are used, they all rely on high-quality object models. In other words, superior object models are a prerequisite for accurate 6D pose estimation. Therefore, this PhD thesis explores a novel 3D vision system from reconstruction and perception aspects.

In the early ages of 3D computer vision, it mainly dealt with polyhedral objects [16, 17]. To recognize fine-grained geometric shapes, some of the methods [18, 19] estimate 6D poses of objects from a 3D point cloud directly. They leveraged point feature description to compute a set of correspondences between the object model and scene point cloud, and also robustly estimate the 6D pose by a random sample consensus (RANSAC) algorithm. Apart from point cloud-based methods, image-based approaches also made significant progress. Hinterstoisser et al. [3] proposed a template matching method, Linemod, which exploits colour gradients and surface normals to describe object contour and interior information. It is further improved in [8] by generating a templates dataset automatically from CAD 3D models. However, those works cannot handle textureless objects in a cluttered environment, due to the lack of rich textures, to detect distinguishable features for matching.

More recently, the emergence of deep convolutional networks techniques, especially CNN-based category detectors [20, 21, 22], have shown excellent outcomes for object detection and object segmentation. Inspired by the remarkable progress on 2D object detection, an increasing number of works employed deep learning for 6D pose estimation. For RGB-derived 6D pose estimation, most approaches follow a similar paradigm: first, they adopt a neural network to detect the eight 3D bounding box corner-points associated with the target objects. They perform a Perspective-n-Point (PnP) algorithm calculating the orientation and translation. However, this paradigm suffers from severe shortcomings in terms of low detection accuracy for texture-less objects and expensive post-processing steps [23].

Compared with 2D images, the point cloud is closer to the original 3D geometric shape and carries a reliable depth distance to localize objects accurately. Qi et al. propose an end-to-end network, PointNet [24], which directly feed point clouds as inputs and performs classification and segmentation for the input point cloud. In [25], an improved version of PointNet was proposed, enabling the network to learn local structures at different scales. While these methods are still at the category level, Li proposed PointRCNN [26], which can achieve 3D object detection from a raw point cloud at the level of instances. However, a typical scene model contains more than a 100k points; training such networks requires high computational and memory requirements. What's more, current point cloud-based learning techniques leverage publicly available datasets to train and test. Although a dataset can speed up the research process, how to generate a generic system that can apply to real-world objects is still challenging.

Overall, the 6D pose estimation is still in its infancy, despite the few frameworks showing some exciting results when evaluated through a dataset. The current 6D pose estimation methods rely heavily on high-quality object and scene models, whether based on RGB image, RGB-D data or 3D point cloud. Because of this, the detection accuracy of those frameworks will decrease significantly when tested in real-world objects. These issues serve as the primary motivation of this thesis - to produce a complete, effective and precise 3D vision system to address core tasks for 3D perception.

1.2 Challenges of 6D Pose Estimation

Although significant progress had been achieved, but several challenges remain to be solved in the field of 6D pose estimation.

First challenge is environmental complexity and clutter. Occlusion and illumination have a significant impact on the accuracy of 6D pose estimation. Occlusion can cause feature missing and thus reduce estimation accuracy. In addition, the intensity of pixels from same part of object are different under various illumination conditions. This factors may result in the system giving faulty or incorrect results.

Second is data generation. Supervised-learning rely on enormous training data, while annotating 6D pose of object in a 3D space is difficult, especially for orientation [4]. Recently, self-supervised 6D pose estimation is attractive, where robots autonomously annotate real world data with accurate 6D object poses [27, 28]. However, self-supervised methods are highly dependent on accuracy of the initial pose and they still need pre-training. What's more, synthetic dataset generators [29, 30] are also widely used for 6D dataset generation.

The third challenge is estimation of non-rigid objects. In real world, a broader class of objects are non-rigid objects such as humans, pants or socks without fixed form [31]. However, 6D pose estimation for non-rigid objects require the more global information extracted from larger receptive fields and are inescapably more sensitive to occlusions.

1.3 Aims and Objectives

This thesis aims to develop a novel 3D intelligent robot vision system that can accomplish object modelling, learning, detection and grasping efficiently. Towards this goal, we propose the following research objectives:

- Objective 1: Implementing a point cloud-based reconstruction system for the textureless and small-scale scene.
- Objective 2: Recovering the accurate 6D pose from different type of input signals.
- Objective 3: Integrating the 6D pose information with ROS system to make the robot to interpret the perceived environment.

1.4 Contribution

This thesis makes the following contributions in the area of the 3D object detection community:

- In Chapter 3, we first propose a real-time, high-quality, 3D scanning pipeline for texture-less scenes. Compared with current state methods, the approach proposed in this chapter can balance speed and quality. The chapter then presents a learning-based algorithm for 6D pose estimation in point cloud data, which exploits neural networks to extract 3D feature description.

- In chapter 4, we propose a novel 6D pose estimation method Image-6D that detects objects, segments instances, and predicts 6D pose simultaneously from a single RGB image without any PnP process. Besides, we introduce Contour-Alignment, an efficient algorithm for pose refinement in an RGB image.

- In chapter 5, we develop Point-6D, a 6D pose estimation neural network using colored point cloud as the input. This network implements 6D pose estimation at a per-point level that improves the performance in an occlusion scene. Point-6D achieves more accurate detection than previous efforts, which fuses 3D geometric data with 2D photometric information.

- In Appendices B, we develop an advanced human-robot interaction system, which leverages 3D vision systems presented in previous chapters as the backbone. This case study shows how those novel methods support robot manipulation in remote handling operations in a hazardous environment.

1.5 Outline of the Thesis

The remaining part of the paper proceeds as follows:

- Chapter 2 describes the current state of 3D object reconstruction and recognition as well as its inadequacies.

- Chapter 3 shows a novel reconstruction pipeline for the small-scale textureless scene.

- Chapter 4 presents a novel 6D pose estimation method Image-6D that can detect objects, segment instances and predict 6D pose of object simultaneously from a single RGB image.

- Chapter 5 introduces a point cloud-based 6D pose estimation technique Point-6D that fully utilizes object geometric and coloured features to regress 6D pose at a per-point level.
- Chapter 6 summarizes the entire thesis and promising future directions in 3D object detection.

In addition, it is worth mentioning that the work in Chapter 3 is published in:

Jin, Y.; A., Soto, D.A.P. ; Rossiter, J. and Veres, S. (2021). Advanced Environment Modelling forRemote Teleoperation to Improve Operator Experience. In Proceedings of the International Conference on Artificial Intelligence (icARTi '21), ISBN 978-1-4503-8575-6/21/12, DOI: 10.1145/3487923.3487939

and the work in Chapter 4 is awarded best paper of ROBOVIS2021 and publised in:

Jin, Y.; Rossiter, J. and Veres, S. (2021). Accurate 6D Object Pose Estimation and Refinement in Cluttered Scenes. In Proceedings of the 2nd International Conference on Robotics, Computer Vision and Intelligent Systems, ISBN 978-989-758-537-1, pages 31-39. DOI: 10.5220/0010654500003061

Chapter 2

Literature Review

This chapter investigates the previous and current technologies related to 3D object reconstruction and detection. Before reviewing the literature, we first give an introduction to the basic concepts of 3D vision. After that, we review existing methods on 3D environment reconstruction from offline and online perspectives. After reviewing the study of 3D reconstruction, the following section discusses the current 3D detection technologies and their intended applications. This review will give the reader a solid base for this thesis developments and contributions as while as a good grounding in the field of application for such 3D vision systems.

2.1 Basic Concepts of 3D Vision

Opposite to 2D vision technology, 3D sensor have ability provide depth information, which is a critical information for many computer vision tasks and image processing applications such as autonomous driving, robotic grasping and augment reality [32]. In the section that follows, we present two essential concepts of 3D vision, which are 3D object representation and 3D data acquisition.

2.1.1 3D Representation

Point Cloud

The point cloud is a set of three-dimension points that can represent the shape of the object or geometric information of the environment. As an ordinarily adopted format, the point cloud is closer to the original geometric shape containing rich feature and scale information without any discretization. However, it is hard to process the raw point cloud directly due to the irregular, high-dimensions and noise of the point cloud data.

Mesh

The mesh consists of plenty of polygons that typically are triangular or quadrangles. The mesh can be converted from a point cloud, and it has more photo realism compared with the point cloud. This type of data is normally used for visualization.

Volumetric Grid

As we mentioned before, the point cloud is irregular and dense, which causes time complexity. A point cloud is typically converted to a volumetric grid by voxelization when passing into the neural network. The volumetric grid is a sample and regular representation. Typically, a volumetric grid divides a 3D space into a set of cells. The voxelization process checks whether or not voxels include the object points and assign a value of 0 or 1.

Depth Map

A depth map is a 2D image that contains the distances from the focal plane to points in the scene. The depth only includes position related data and miss colour information. Therefore, it is generally used with colour images together, commonly referred to as RGB-D data.

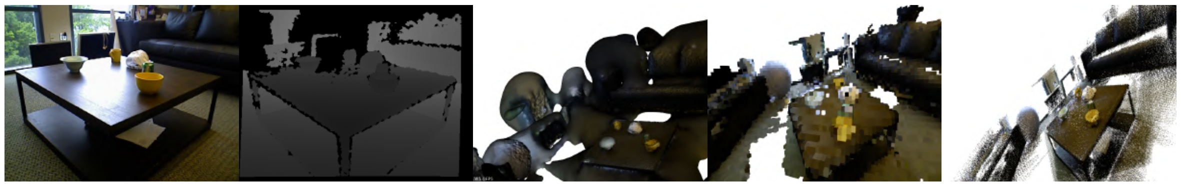

(a) Color Image (b) Depth Map (c) Mesh (d) Volumetric (e) Point Cloud

Figure 2.1: Several different 3D representations.

2.1.2 3D Acquisition

The 3D acquisition methods can be divided into two core classes, active and passive sensors. Specifically, monocular and stereo cameras are passive sensors, while active measurement encompasses structured light and Time of Flight (TOF). Principles of those approaches are described in detail in the following section, and their benefits and drawbacks are compared.

Monocular 3D vision

The monocular camera can only capture 2D colour image information and cannot obtain 3D data directly. Recovering the 3D scene model is usually combined with a Structure from Motion (SfM) technique, which captures several monocular images from various viewpoints to compute the corresponding camera pose. This monocular 3D vision is regarded as a subcategory of multi-view 3D object reconstruction. In addition, single-view 3D reconstruction from a single monocular image also made excellent progress [33]. It primarily utilizes deep learning networks and prior knowledge of shapes. However, this approach can only recover individual objects at the moment and cannot obtain 3D understanding of the entire scene.

Stereo 3D Vision

In this thesis, the term 'stereo vision' refers to binocular stereo vision, which utilizes the parallax of two cameras to compute depth information. Specifically, a stereo camera can acquire two images and calculate the location deviation between the corresponding points of two images to obtain the 3D geometric data of the current scene view. The stereo camera is low-cost and can be used both indoors and outdoors. What's more, it can generate a dense per-pixel depth map that provides fast and accurate 3D data. However, this device also has some obvious disadvantages. It is sensitive to illumination because light changes cause pixel loss. On the other hand, this acquisition method is purely visual and algorithmically demanding. Currently, a widely-used stereo camera is the ZED camera, which can acquire an immense depth range from 0.3 to 25m and can provide a maximum of 2K video resolutions.

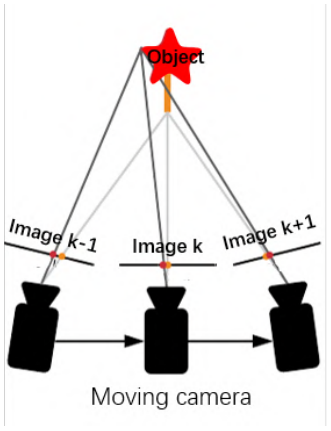
Structured Light

The structured-light system explores an active projection device to measure the depth distance. As an infrared ray projected onto the object with a particular encoding, it deforms, allowing vision systems to measure the depth and surface details of the objects by calculating the deformities of the captured reflected light pattern. Depending on the coding pattern, structured light can be divided into two main categories: strip indexing and grid indexing. The advantage of the structured-light sensor is that it can work on dark light, and the algorithm is proven. But the acquisition accuracy will decrease a lot for the outdoor environment and large-scale scene. Some notable structured light 3D sensors, such as the Kinect-1, Inter RealSense, etc., are extensively used in 3D vision research.

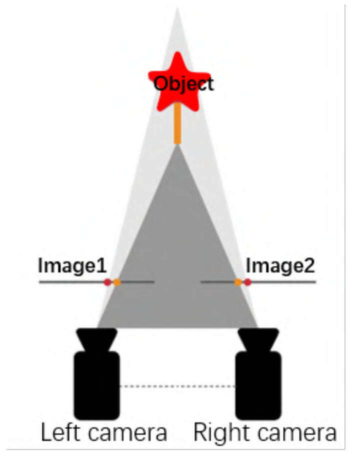
Time of Flight

As the name suggests, the Time of Flight (ToF) measures light flight time to obtain the distance between sensor and object. Specifically, the sensor emits a continuous laser pulse for the object and then receives the reflected light, and the distance is calculated by the time of flight multiplied by the speed of light. Because direct measurement of flight time is complex, the ToF sensor usually adopts modulated light and measures the reflected light's phase.

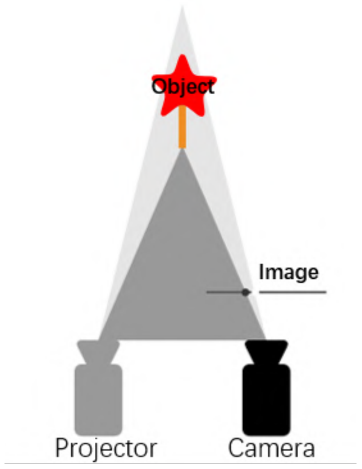
According to the modulation type, TOF methods can be divided into two general categories: Pulsed Modulation and Continuous Wave Modulation. Pulse modulation requires very high precision clocks for measurement, so most ToF cameras adopt the latter. Although expensive, ToF-based LiDAR has a dominant position in autonomous driving due to high precision measurement. What's more, Kinect v2, SoftKinetic and other ToF devices are widely used in numerous 3D detection tasks.



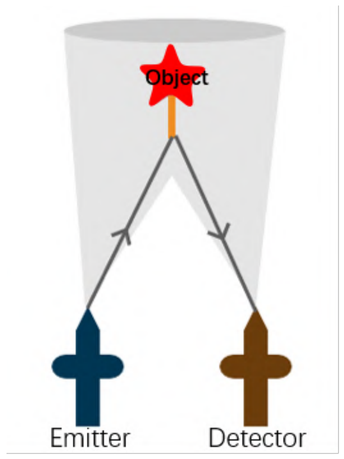
(a) Monocular 3D Vision



(b) Stereo Camera



(c) Structured Light



(d) Time of Flight

Figure 2.2: Several different 3D acquisition methods

2.1.3 6D Pose Representation

This section introduces a group of preliminaries that regards this thesis. The 6D is the abbreviation for 6 degrees of freedom, which is composed of the 3D translation and the 3D rotation in each of the 3D axis. In the rest of thesis, the symbol $\boldsymbol{t}$ denotes the 3D translation and $\mathbf{R}$ represents the 3D rotation. Normally, there are three different types of 3D rotation: (1). euler-angle, (2). rotation matrix and (3). quaternion, and they are interchangeable.

Euler-angle describes the magnitude of the rotation about the axis in a three-dimensional Euclidean space and denotes as (α, β, γ). The range of a axis-angle is from 0 ° to 360 °. The 3D rotation matrix is a 3×3 transformation matrix, which performs the rotation of vectors in Euclidean space. The relationship between axis-angle and rotation matrix is:

$$\begin{aligned} R &= \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} \\ &= \begin{bmatrix} cos\alpha cos\beta & cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma & cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma \\ sin\alpha cos\beta & sin\alpha sin\beta sin\gamma + cos\alpha cos\gamma & sin\alpha sin\beta cos\gamma - cos\alpha sin\gamma \\ -sin\beta & cos\beta sin\gamma & cos\beta cos \end{bmatrix} \end{aligned} \tag{2.1}$$

Apart from axis-angle and rotation matrix, quaternion is also a typical rotation representation in 3D space. A quaternion is a 4×1 unit vector $q = [x, y, z, w]$ where x,y and z are vector part and w is the scalar part. The conversion between quaternion and rotation matrix is:

$$R = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy + 2wz & 2xz - 2wy \\ 2xy - 2wz & 1 - 2x^2 - 2z^2 & 2yz + 2wx \\ 2xz + 2wy & 2yz - 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix} \tag{2.2}$$

The 3×3 rotation matrix $\boldsymbol{R}$ and 3×1 translation matrix $\boldsymbol{t}$ composes transformation matrix $\{\mathbf{R}|\mathbf{t}\}$, denotes as $\boldsymbol{T}$. In the point cloud registration, the transformation matrix

$\boldsymbol{T}$ describes the position of the source point cloud origin and the orientation of its axes, relative to the frame of target.

2.1.4 Datasets

A dataset is a collection of data that contains a lot of separate pieces of information. Using a standard dataset is beneficial for training, testing and evaluating an algorithm in the research. In this thesis, we employ two main classes of datasets: one is visual-SLAM dataset like RGB-D Scenes v.2 [7] and the SUN3D [12] Dataset, and another is 6D pose estimation dataset, such as LineMOD [8], LineMOD-OCCLUDED [8] and YCB-Video [4] dataset.

The visual-SLAM dataset contains scene data: the RGB image, depth image and point cloud, and ground truth information that is the pose matrix of the scene. 6D pose estimation dataset includes scene data and objects instance mask, and its ground truth information is the pose matrix of each object. We will detail those datasets when they are used in later chapters.

2.2 3D Environment Reconstruction

Nowadays, 3D reconstruction technology has become a key instrument in various fields, exemplified by 3D printing, robotics, architecture, and augmented reality fields [34]. There are three fundamental methodologies used in modelling unknown environments: SfM, visual simultaneous localization and mapping (Visual-SLAM), and deep learning-based reconstruction. In this section, the typical SfM/Visual-SLAM pipelines and learning-based methods are reviewed, with a strong emphasis on recent work on real-time online reconstruction approaches.

2.2.1 Structure from Motion

Structure from Motion (SfM) is a prevailing technique to reconstruct 3D objects through processing multi-view 2D images [35]. A classical SfM method can be outlined in five steps: feature extraction, building image correspondences, camera pose estimation, calculating the position of 3D points, and non-linear optimisation shown in Figure 2.3 [36].

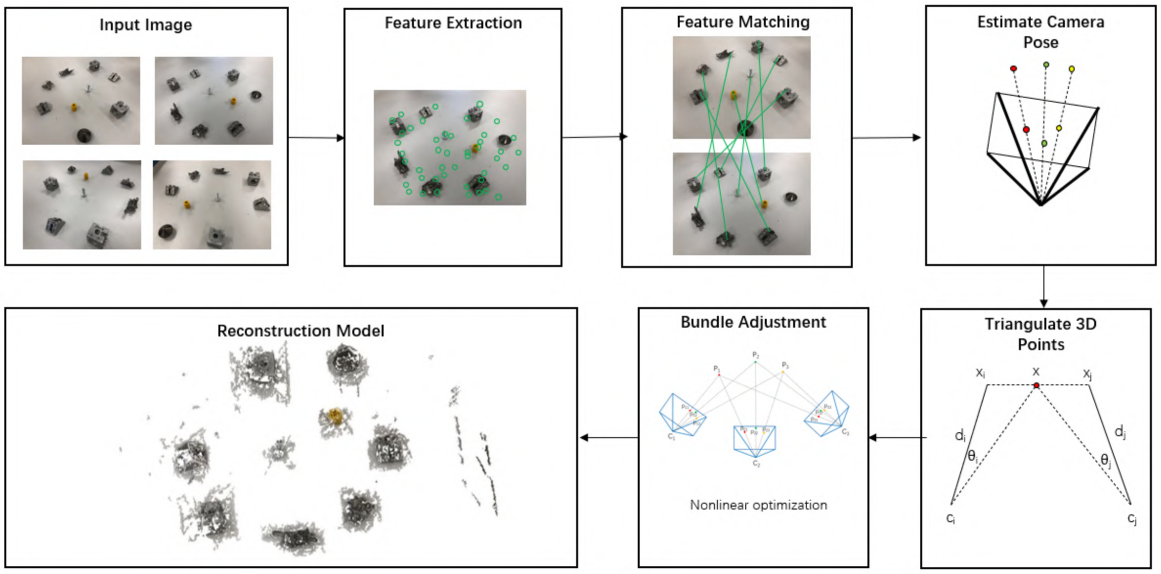


Figure 2.3: A general pipeline for SfM.

Firstly, the SfM system extracts features [37, 38, 39, 40] from images, of which scale-invariant feature transform algorithm (SIFT) [37] and speeded up robust features algorithm (SURF) [38] are two of the most well-known feature extraction algorithms. Secondly, SfM measures the similarity between those feature descriptors by calculating the Euclidean distance between each feature point and generates a set of potential matches. Following that, RANSAC [41] is used to remove outlier matches, and estimate camera poses matrices [42] between pairs of images. Then, the initial 3D structure can be recovered by triangulation [43] that calculate the error between measurement and projected 3D point. The final step is bundle adjustment (BA) [44] that refines the camera poses, 3D points, also intrinsic camera parameters together.

Depending on the image addition sequence in the SfM process, existing SfM algorithms can be classified as incremental [45, 35, 46], hierarchical [47, 48], and global approaches [49, 50, 36]. Incremental SfM greedily adds one image at a time and solves bundle adjustment repeatedly. The incremental approach is robust for outliers and works well with large collections of images, providing remarkably accurate reconstructions in many circumstances. However, an incremental approach requires a good initial matching pair that contains a sufficient number of matching points and sensitivities to the sequence of addition, which can cause an accumulated error. Obviously, global methods, which calculate all images simultaneously, don't need to consider the initial image and addition sequence. Moreover, the global method takes one bundle adjustment process and is highly efficient for reconstruction. Hierarchical methods [47, 48] integrate advantages of global and incremental to tackle the problems of efficiency and accuracy.

Overall, SfM is a typical offline method that can reconstruct a stationary scene precisely. However, the most significant limitation of SfM is that this algorithm relies substantially on scene features and requires a long calculation time.

2.2.2 Visual-SLAM

Simultaneous Localization and Mapping (SLAM) is a method to get the camera motion and the 3D structure of the scene in an unknown environment in real-time. The input of SLAM is usually a visual signal, so it is also referred to as visual-SLAM. Compared with the SfM technique, the visual-SLAM algorithm can achieve real-time environment reconstruction and sensor pose estimation. Because of this, visual-SLAM is widely used in the field of augmented reality (AR), unmanned autonomous vehicles (UAV) and telepresence.

Generally, the visual-SLAM is comprised of five modules: (1) Initialization; (2) tracking; (3) mapping; (4) relocalization; and (5) global map optimization. In the initialization, the system will determine a globe coordinate system and generate an initial map in this coordinate system. Then, tracking is to calculate the camera pose of the current frame through computing the correspondences between the image and recontoured map. This step involves solving the PnP [42] problem that is also used in SfM. In the mapping process, the reconstructed map is updated by a tracked camera pose. The following two modules, relocalization and global map optimization, are used to refine the reconstructed model. Specifically, relocalization is to recover the camera pose again when tracking is lost. As for the global maps, the optimization module is used to minimize the accumulative estimation error of camera movement. The reference information can be obtained firstly by matching the initial map with the current image. Then, a loop constraint is employed as a restriction to minimize the error.

As mentioned above, the five core components of visual-SLAM are adopted by most algorithms, although each method employs a different strategy for each module. In this section, the review of the visual-SLAM algorithm is categorized according to the type of input data.

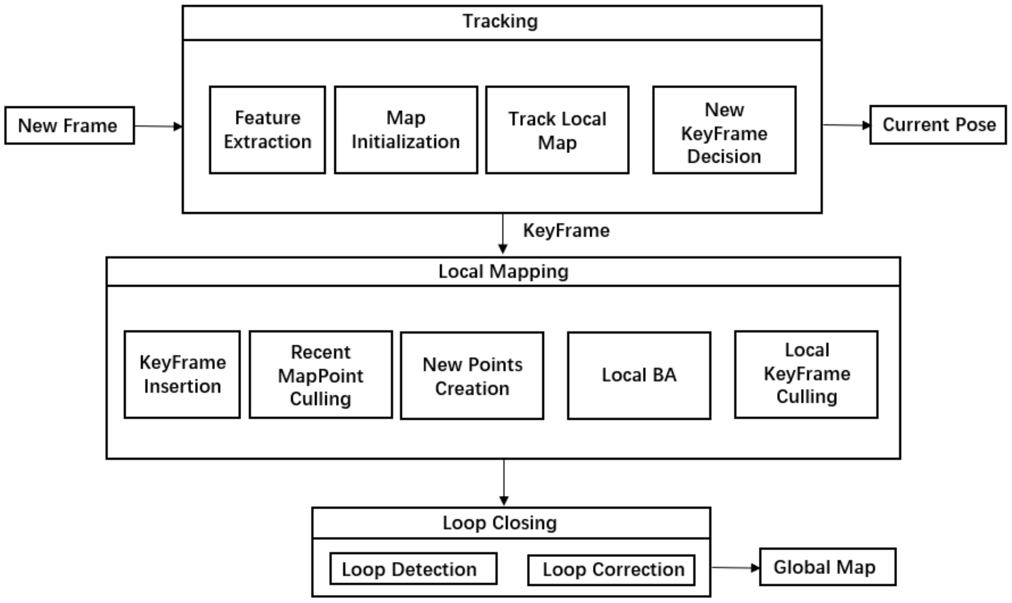
Monocular Image-Based Visual-SLAM

The early research of visual-SLAM utilises a monocular camera as an input device. Davison et al. [51] proposed the first monocular visual-SLAM system MonoSLAM, which exploits an extended Kalman filter to predict camera motion and update the state. However, the size of a state vector becomes large with the scene extending, making monoSLAM less appropriate to reconstruct a large-scale environment due to the increased computational demands. To overcome the challenge of large-scale reconstruction and speed up the reconstruction process, PTAM was developed by Klein and Murray [52]. PTAM exploits multi-thread executing tracking and parallel mapping. Note that the 3D points are estimated by triangulation and then optimised by BA in PTAM, which uses the same method as the previous SfM pipeline. Inspired by Mur-Artal et al. [5], researchers proposed a real-time and accurate monocular visual-SLAM system ORB-SLAM. In this work, the author used the ORB feature [39] detector to replace the previous SIFT [37] and SURF [38] feature detector to accelerate feature extraction. What's more, ORB-SLAM improves many of the shortcomings of PTAM, such as lack of camera relocalisation and loop closing. This limits the usefulness of resulting map reconstructions, which cannot be used in interactive robotics tasks or advanced augmented reality applications. As Figure 2.4 (a) shown, The ORB-SLAM pipeline composes tracking, mapping, relocalization, and loop closing components that improves robustness and stability of system.

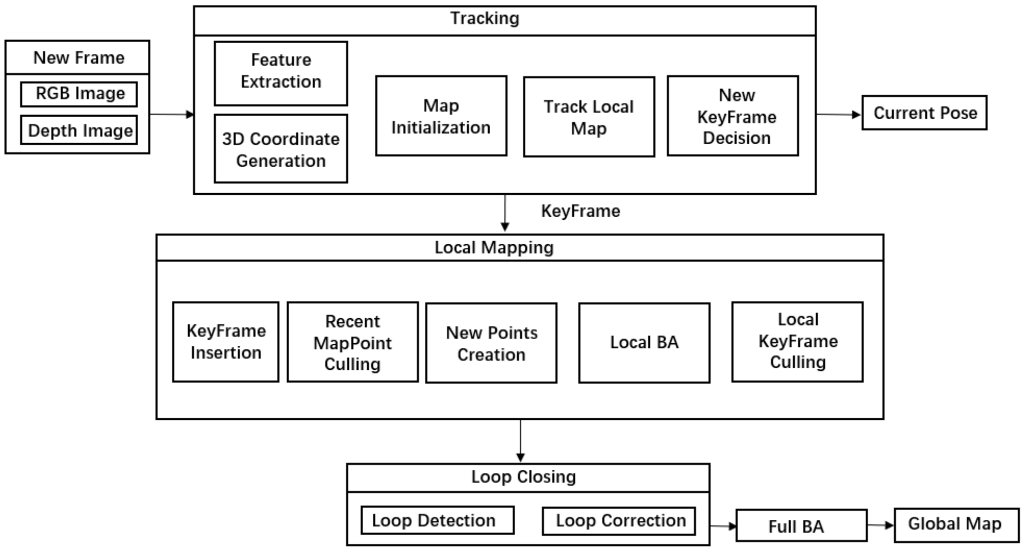
Although those works achieved remarkable progress on environment reconstruction, they can only output the sparse map, which cannot be applied in interactive robotics works or advanced AR applications. Recently, with the rapid growth of GPU computing, there are several kinds of research focusing on real-time dense monocular reconstruction, such as DeepFusion [53] and DeepFactors [54], typically generate dense scene maps by estimating the depth values from a single RGB image. Unfortunately, recovering the depth information from the monocular image is not precise due to the presence of occlusions, noise and repeated texture. Overall, RGB-D camera-based visual-SLAM may be a better choice for dense environments modelling.

RGB-D Image-Based Visual-SLAM

In Section 2.1.2, we introduced a variety of RGB-D cameras, which have been widely applied in visual computing. For example, the seminal KinectFusion system [55] had a significant effect in the field of 3D reconstruction. In this study, a novel GPU-based pipeline is implemented to calculate camera motion from scene motion. KinectFusion utilized Truncated Signed Distance Functions (TSDFs) [56] representing 3D scenes, and a significant advantage of TSDFs is that depth images at any viewing can be efficiently obtained compared with Mesh methods. For estimating the pose of the camera, KinectFusion used Iterative Closest Point (ICP) [57] algorithm. However, there are certain drawbacks associated with using the ICP algorithm, which is burdensome computation when the model involves a tremendous amount of cloud points. Subsequently,



(a) Overall of ORB-SLAM framework [5]



(b) Overall of ORB-SLAM2 framework [6]

Figure 2.4: A comparison between RGB-based [5] and RGBD-based [6] visual-SLAM

Whelan et al. [58] developed the Kintinous algorithm based on KinectFusion, which benefits in increasing computing power through integrating fast odometer from visions (FOVIS) into the GPU pipeline.

However, both KinectFusion and Kintinous are unable to handle loop closures, and as a result, may drift indefinitely. RGB-D SLAM [59] overcomes this problem by integrating multiple motion estimation and additionally estimating the motion to frames other than the direct computation. Moreove, RGB-D SLAM may be the first to take advantage of the colour image and depth maps. Compared with KinectFusion which only rely on depth information, RGB-D SLAM exploits colour images to extract feature and depth information for validation. The recent lightweight visual-SLAM system ORB-SLAM2 [6] utilized bundle globe adjustment (BA) replacing the previous ICP algorithm to achieve higher accuracy of motion estimation as illustrated in Figure 2.4 (b). Dual to ORB-SLAM2 suffering from many tracking failures, Dai et al. [60] proposed a real-time and high-quality reconstruction framework BundleFusion. In this work, he designed a hierarchical local-to-global pose optimization approach. On the first hierarchy level, every n consecutive frames compose a chunk, and the first frame is defined as the keyframe. Then, it performed a local pose optimization for a chunk. Finally, the keyframes of all chunks are globally optimized. Choi et al. [11] noticed that pairwise registration is prone to errors in global registration. So they proposed a robust global optimization based on line processes to improve the reconstruction quality of indoor scenes.

Point Cloud-Based Visual-SLAM

As an alternative to image intensity, the point-based strategy is employed by several reconstruction methods. For the point-based visual-SLAM systems, the core problem is multiview point cloud registration. Point cloud registration is the process of aligning two or more 3-D point clouds of the same scene into a standard coordinate system.

The first registration group is the local registration, such as the multiview ICP-like scheme and 3D point correspondences. ICP-based methods mainly utilized the ICP processing by solving the least-squares problem to obtain a camera pose [61, 62, 10]. With regard to the point matching approaches, they were concluded into four phases: (1) 3D keypoint feature detection; (2) correspondences matching; (3) outlier rejection and (4) transformation calculation. Figure 2.5 illustrates a general pipeline for point cloud registration.

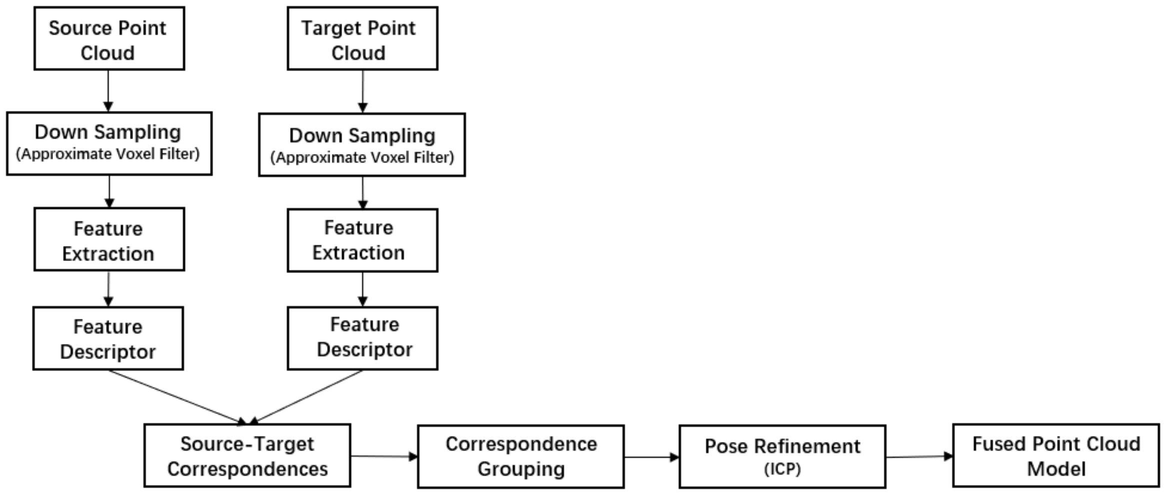


Figure 2.5: A general pipeline for point matching approaches.

The common 3D keypoint detection methods included SIFT-3D [63], ISSKeyPoint-3D [64] and Harris-3D [65] etc. What is more, a variety of feature descriptors have been proposed to describe geometrical patterns based on the information around the key points, such as Signature of Histograms of OrienTations (SHOT) [66], Fast Point Feature Histograms (FPFH) [67], Camera Roll Histogram (CRH) [68] descriptors and so on. In correspondences matching, k-d tree [69] and octree [70] are normally employed to search the point correspondences. The searching result included several mismatching correspondences which could lead to false pose estimation. To reject outliers, [18] proposed a 3D hough voting algorithm and Buch et al. [19] developed a pre-rejective RANSAC method as verification. However, the RANSAC-based registration [18, 19] requires a long time for calculation and evaluation. Consequently, Zhou et al. [71]

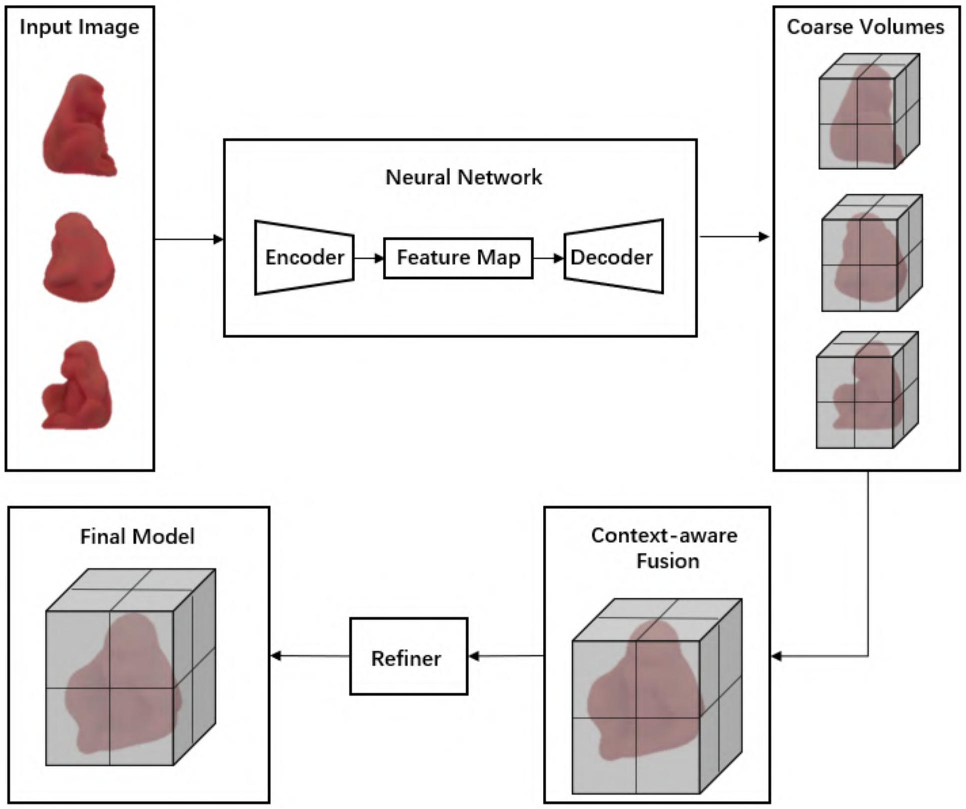
proposed a faster registration algorithm that quickly optimizes a robust objective of a few correspondences.

A majority of local registration approaches endure the growing complexity of correspondence computation. To get rid of it, some recent approaches [11] make use of the global registration pipeline, which does not need initialization and iterative sampling. Note that the global registration methods are faster than the local refinement algorithm because they do not require recomputing correspondences. However, this strategy comes at the expense of accuracy.

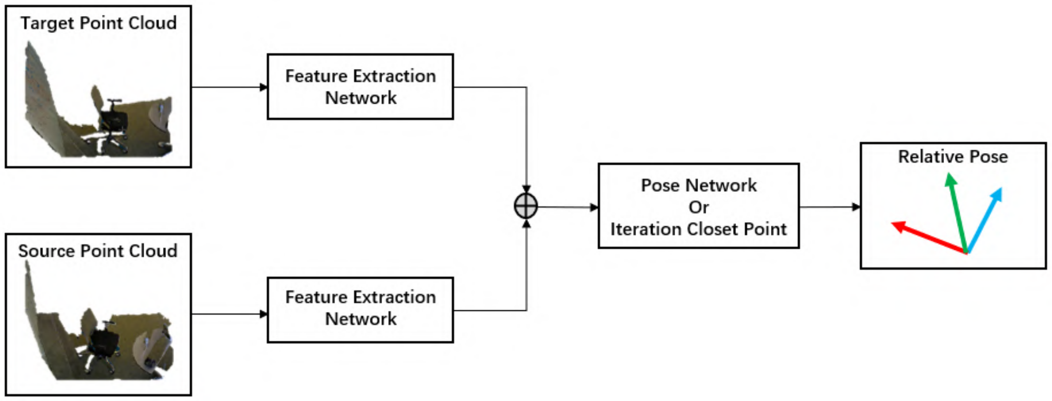
Deep Learning-Based Reconstruction

In the last decade, learning-based 3D reconstruction using CNN has attracted increasing research interests and proved an exciting achievement. Recent neural networks based methods recover a complete 3D scene model by learnt features from multiple inputs. Eigen et al. [72] and Laina et al. [73] successively predict depth maps from a single colour image by using a convolutional neural network. Note that the traditional methods, such as the previously mentioned SfM and visual-SLAM, fail in single view reconstruction. Subsequently, 3D-R2N2 [74] achieved both single- and multi-view reconstruction in a single network. The 3D-R2N2 is established based on the standard Long Short Term Memory (LSTM) network [75] that can keep the previous iteration and incrementally improve the output result.

Inspired by the success of 3D-R2N2, Learnt Stereo Machines (LSM) [78] and 3D2SeqViews, Han et al. [79] continued using LSTM architecture and improved reconstruction accuracy by exploiting more geometric cues. The major problem with 3D-R2N2 and LSM is that the order of the input image influences the reconstruction quality due to the LSTM unit is a permutation variant. Also, LSTM is time-consuming because the input data are feed sequentially without parallelization. To address the above problem, 3DensiNet [80] employs a maximum pooling layer to gather cues from multiple images, and Pix2Vox [76] built multiple encoder-decoder architectures that can run in parallel.



(a) The architecture of Pix2Vox [76], which can recover objects mesh exploiting image from single or multiple view. The object image is from LineMOD dataset[8].


Target Point Cloud
Feature Extraction Network
Source Point Cloud
Feature Extraction Network
Pose Network Or Iteration Closet Point
Relative Pose

(b) Overall of Deng et al. [77]'s framework, which utilize PPF-FoldNet and PC-FoldNet to as feature extraction to predict local descriptors, then sent the matched features to a RelativeNet to estimate relative pose.

Figure 2.6: Two typical learning-based neural network architectures for reconstruction. Top is voxel-based approach and bottom is point cloud-based method. The point cloud image is from RGB-D Scenes Dataset [7]

For the above methods, the 3D shape of the reconstruction result is represented by a 3D voxel grid shown in Figure 2.6 (a), so we call them voxel-based methods. Although voxel-based approaches gain from recovering 3D shapes, they rely heavily on computational power and are mainly specific for single object reconstruction rather than scene reconstruction. Another direction of learning-based reconstruction is processing point cloud data directly.

Recently, numerous works are starting to leverage deep neural networks to learn local features from a point cloud since Zeng et al. [81] proposed their pioneering work, called 3DMatch, which is a 3D convolutional neural network. Closely related to 3DMatch, Yew and Lee [82], Choy et al. [83] and Deng et al. [77] respectively developed an effective neural network to extra the 3D local descriptors. In addition, Ao et al. [84], Khoury et al. [85] and Deng et al. [86] tried to compute rotation invariant descriptors which are more robust for the scenario with strong rotational changes. The Figure 2.6 (b) shows the architecture of Deng et al. [77]' network, which use PPF-FoldNet [86] and PC-FoldNet to extra local descriptors from input point cloud and then feed the matched features to a RelativeNet to estimate relative pose of source and target point cloud.

Apart from using a neural network to extract features, Choy et al. [87] propose a 6-dimensional convolutional network that can predict a set of inliers correspondences between the source feature $F_x = \{f_{x_1}, ..., f_{x_{N_x}}\}$ and target feature $F_y = \{f_{y_1}, ..., f_{y_{N_x}}\}$. In addition, Pais et al. [88] proposed 3DRegNet used to distinguish inliers and outliers from point correspondences.

However, the performance of learning-based methods drops a lot when they deal with unseen scenes. As a result, the one goal of this thesis is to develop a fast and accurate point cloud reconstruction method. We eschew the deep learning networks and instead fully exploit the geometric and photometric information of the point cloud itself to achieve this goal.

2.3 6D Objects Pose Estimation

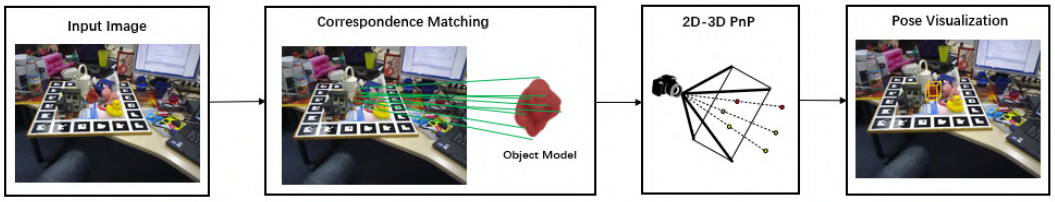
6D pose estimation is a task of predicting the 6D pose of an object in the camera coordinate system, which encompasses its location and orientation. The significance of 6D pose estimation is to obtain the precise pose of the object, which is mainly applied in the field of robot grasping, augmented reality, and autonomous vehicles [15]. A 6D pose estimation task can be divided into 2D image-based and 3D point cloud-based methods according to the input data used. Specific to the implementation method, it can be further divided into a correspondence-based, a template-based and a voting-based strategy [89]. An overall comparion of 6D pose estimation methods has listed in Table 2.1.

2.3.1 Image-based 6D Objects Pose Estimation

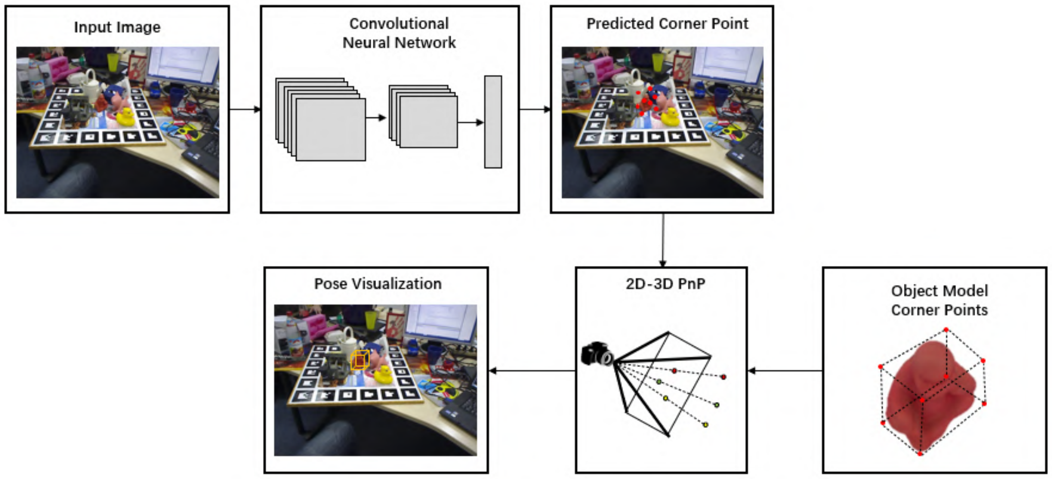
Correspondence-based 6D Objects Pose Estimation

Correspondence-based methods usually start by establishing 2D-3D correspondence and then leverage a RANSAC-based PnP algorithm or its variant to calculate object poses as Figure 2.7 shown. The traditional correspondence-based approaches mainly utilize SIFT [37] and SURF [38] algorithms for feature extraction, which is similar to the SfM we mentioned before. Sadran et al. [90] propose a sparse SIFT-based 6D object estimation system that reduces the runtime of object pose prediction through removing outliers keypoints. What's more, Grundmann et al. [91] apply this 2D-3D strategy into a robotic manipulation task in a cluttered environment. However, the traditional correspondence-based methods fully rely on the texture information of objects. So, they are unreliable for textureless objects.

The emergence of deep learning techniques, especially CNN-based category detectors, have shown excellent outcomes for object detection [92, 20] and object segmentation [93]. In recent years, there has been an increasing number of 6D pose estimation works which involve the use of CNN [94, 14, 95]. Figure 2.7 (b) indicates a popu-



(a) Transitional correspondence-based method



(b) Learning-based correspondence-based method

Figure 2.7: Typical correspondence-based 6D pose estimation method.The object image is from LineMOD dataset[8]

lar computational paradigm of learning-based approaches in detecting 3D bounding box vertices for objects using CNN and then computing 6D pose by solving the PnP problem. We can notice that the difference between traditional and learning-based methods is feature extraction. Learning-based methods, e.g. BB8 [94], Yolo6D [14], leveraged neural networks to detect eight 2D projection corners. With a similar idea, DOPE [96] detects nine belief maps, that is, one for centroids and eight for corners, and eight vector fields containing the direction from 8 corners to the corresponding centroid. This method can improve detection accuracy compared with a direct output of eight corners.

Hu et al. [97] observed that the performance of previous works is fast and robust but terrible for heavily-occluded and poorly-textured objects. To overcome this challenge, Hu et al. [97] proposed a segmentation-driven 6D pose estimation algorithm, in which each visible component of the object provides clues for 2D projection corners detection. As we mentioned before, correspondence-based methods are a two-stage pipeline, so they are not end-to-end trainable. Wang et al. [98] developed a GDR-Net, which includes a learnable Patch-PnP framework to improve the performance of direct 6D pose regression. However, correspondence-based methods are suffered from either occlusion or low visibility.

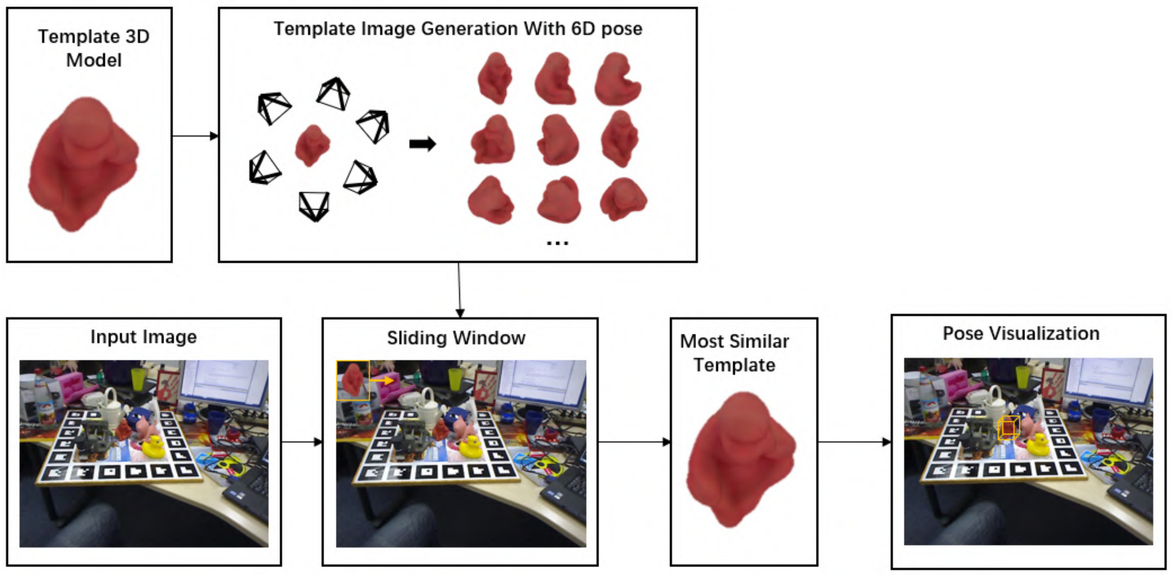
Template-based methods

Figure 2.8 (a) demonstrates a general pipeline for traditional template-based methods. These approaches rely on comparing the similarity between template library and input image to obtain the object 6D pose. Probably the most famous early method is Linemod [3], which calculates a cluttered object's 6D pose exploiting both colour gradients and surface normals. It is then improved by automatically generating templates covering a full view hemisphere in [8]. Hodaň et al. [99] applied a cascade-style evaluation of window position in a template-based method, reducing the computational complexity of sliding window significantly. In short, the template-based method is more robust than correspondence-based approaches for textureless objects, but it leads to low pose detection accuracy in environments full of occluded objects.

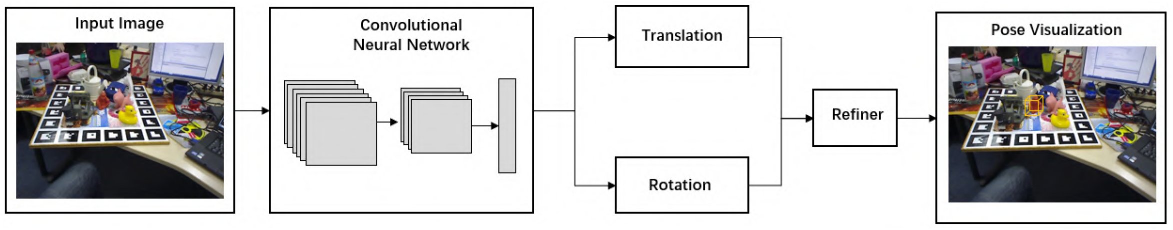
The development of deep learning networks brings new ideas to template-based 6D object estimation tasks. There are numerical works which regress a 6D pose of the target object directly from the input image via supervised learning as Figure 2.8 (b) displayed, which SSD-6D [13], PoseCNN [4], Deep-6DPose [96] et al. are widely used in various robotic tasks. The SSD-6D [13] algorithm employed an SSD network to detect classification, 2D bounding box and 3D rotation of target object, and then, network outputs together with pre-computed information to estimate the object's 3D pose. Inspired by SSD-6D [13], PoseCNN [4] and Deep-6DPose [96] additionally detects the mask of instance to improve the accuracy. For pose estimation, Deep-6DPose [96]

Table 2.1: A comparison of state-of-arts 6D pose estimation methods

Data Source	Methods		Is DL based
RGB	Correspondence-based	Sadran et al. [90]	×
		Grundmann et al. [91]	×
		BB8[94]	√
		YOLO6D[14]	√
		DOPE[96]	√
		GDR-Net[98]	√
	Template-based	Linemod[3, 8]	×
		SSD-6D[13]	√
		PoseCNN[4]	√
		Deep-6DPose[96]	√
	Voting-based	Brachmann et al. [100]	×
		Dong et al. [101]	×
		PVNet[23]	√
Point Cloud	Correspondence-based	3DMatch[81]	√
		PPFnet[86]	√
		3DFeat-Net[82]	√
		Rpm-net[102]	√
	Template-based	Frustum PointNets[103]	√
		PointFusion[104]	√
		DenseFusion[15]	√
		CloudPose[105]	√
	Voting-based	Tombari and Di Stefano [18]	×
		Woodford et al. [106]	×
		Yoloff [107]	√
		PVN3D[9]	√
		VoteNet[108]	√
		ImVoteNet[109]	√



(a) Traditional template-based method



(b) Deep learning-based method

Figure 2.8: Template-based object 6D pose estimation methods.The object image is from LineMOD dataset[8]

use Lie algebra to represent rotation. However, an issue for Deep-6DPose is that it struggles for small or symmetrical objects due to exploiting the ROIs from RPN. To overcome it, PoseCNN [4] proposed a novel neural network architecture composed of a feature extraction backbone, embedding step and regression. The 3D translation of a target object is computed according to its centre and distance from the camera. In comparison, the 3D rotation of a target object is obtained through regressing to a quaternion representation. Inspired by PoseCNN, Capellen et al. [110] proposed a parallel network architecture based on PoseCNN, which can detect the rotation for multiple objects at the same time separating from the translation regression.

In recent years, apart from learning using handcraft data, self-supervised 6D object pose estimation is also a popular research field. Training a robust network needs a massive amount of label data, which is hard to scale to all kinds of objects. Deng et al. [27] pioneered a novel self-supervised 6D object pose estimation framework for robotic manipulation. However, it only runs an existing 6D pose estimation to obtain a reliable 6D annotation, and final results still rely on full-supervised learning. To improve it, Wang et al. [28] directly employ self-supervision for 6D pose estimation through enforcing visual and geometric alignment. Note that this method requires pre-training with synthetic RGB data.

One major drawback of this approach is that it requires a precise target object model as a template or source of synthetic training data. Additionally, post-processing such as ICP is usually necessary to refine the pose result.

Voting-based methods

Voting-based methods are designed to solve object detection under severe occlusions and cluttered scenes. As Figure 2.9 illustrated, voting-based methods have a voting verification that key points accumulated enough voting can be used in the next step. Brachmann et al. [100] developed a Vitruvian Manifold algorithm that each pixel in the image votes for a continuous frame in a canonical pose. However, it only achieves pose detection for a single object. For instance, level detection, Zhang and Cao [111] integrate [112, 113] proposing a four-stage 6D pose estimation framework that calculates a rough 6D pose hypothesis using a hash voting-based system. Similar to this, Dong et al. [101] develop a global pruning algorithm to improve voting and learning efficiency. Nevertheless, the above methods slightly enhance the detection quality in an occlusions environment.

An efficient and widely used method is proposed by Peng et al. [23], Pixel-wise Voting Network (PVNet), which detect vectors between pixel and keypoints rather than directly regressing keypoints. Those vectors then vote for keypoints and execute

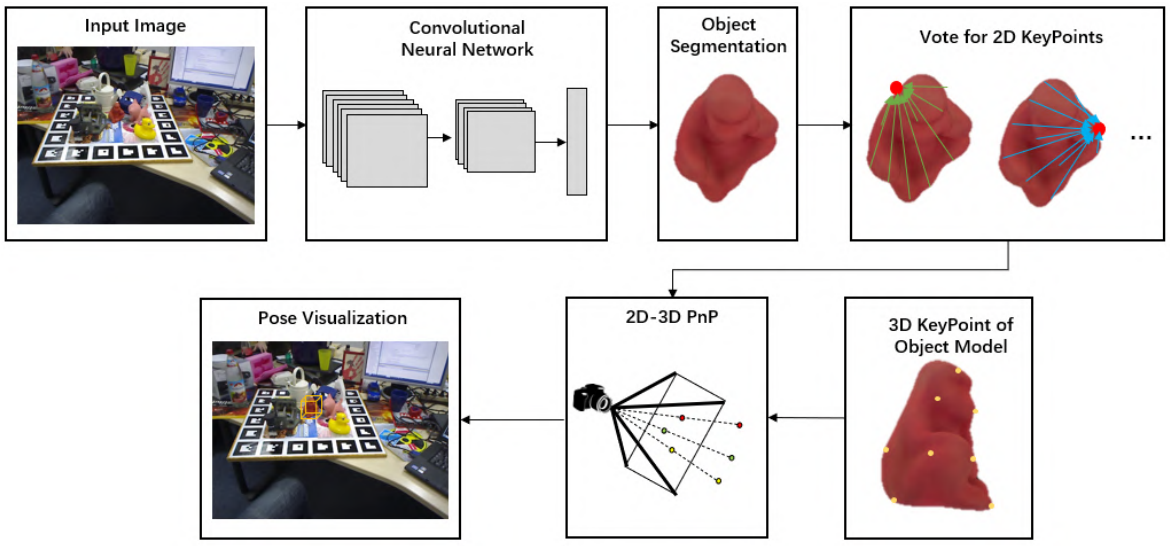


Figure 2.9: Overview architecture of vote-based 6D pose estimation. The object image is from LineMOD dataset [8].

RANSAC. Essentially, it is a variant of the keypoint-based method. Still, the detection performance on occlusions and in cluttered environments improved a lot because it can infer the relative direction to obscured parts according to appearing parts.

2.3.2 Point Cloud-based 6D Objects Pose Estimation

A 3D point cloud contains abundant geometric, shape and scale information, which can provide an opportunity for a richer feature in the cluttered environment for the neural network. With the speedy development of 6D pose estimation, the 3D point cloud-based deep learning is increasingly attracting researchers' attention. In this section, we review the development of point cloud-based 6D objects.

Correspondence-based methods

The main paradigm of point cloud-based 6D pose estimation is similar with image-based as Figure 2.10 indicated, and the difference lies in the feature descriptor. The common 3D keypoint detection methods include SIFT-3D [63], ISSKeyPoint-3D [64] and Harris-3D [65] etc,. What is more, a variety of feature descriptors have been proposed to describe geometrical patterns based on the information around the key points, such as Signature of Histograms of OrienTations (SHOT) [66], Fast Point Feature Histograms (FPFH) [67], Camera Roll Histogram (CRH) [68] descriptors and so on.

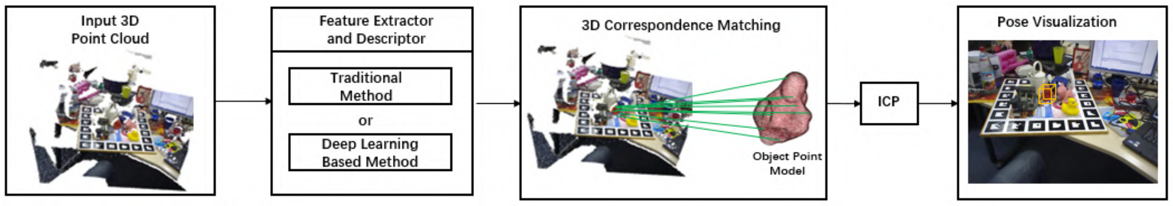

Input 3D Point Cloud
Feature Extractor and Descriptor
Traditional Method
or
Deep Learning Based Method
3D Correspondence Matching
Object Point Model
ICP
Pose Visualization

Figure 2.10: Block diagram of 3D correspondence-based 6D pose estimation methods.The object image is from LineMOD dataset [8].

The transitional 3D feature descriptor works well for 3D point clouds with complete surfaces but is unstable for low-resolution and noisy scenes. Inspired by the recent success of 2D-based 6D estimation, Zeng et al. [81] proposed a 3D convolutional neural network (3DMatch) to learn the object feature descriptor. The 3DMatch system feeds in the local volumetric region around a random keypoint and output a 3D feature descriptor for that point. The author verifies that this method can successfully be applied in object pose estimation, 3D reconstruction, and surface correspondence. However, 3DMatch relies on dense local grids ignoring raw 3D data, its sparsity and unstructured-ness. Deng et al. [86] try to overcome the above challenges by developing Point Pair Feature network (PPFnet), which learns local feature descriptors on pure geometry and fuses features with points and normals to improve tolerance to rotations. In addition, Khoury et al. [85] and Elbaz et al. [114] presented Compact Geometric Features (CGF) and LORAX, respectively, which are low-dimensional feature descriptors to describe local geometry around a keypoint in an unstructured point cloud.

These methods, despite their outstanding quality, can't learn to detect keypoints. Therefore, Yew and Lee [82] investigated 3DFeat-Net that learns both 3D keypoints and feature descriptors for an unstructured point cloud using PointNet [24], whilst

3DSmoothNet [115] and DeepVCP [116] exploits 3D CNNs. Inspired by those works, robust point matching network (Rpm-net) [102] design a hierarchical network operating directly on a point cloud, which makes 3D keypoint and feature descriptors more robust to point clouds compared with 3DFeat-Net [82].

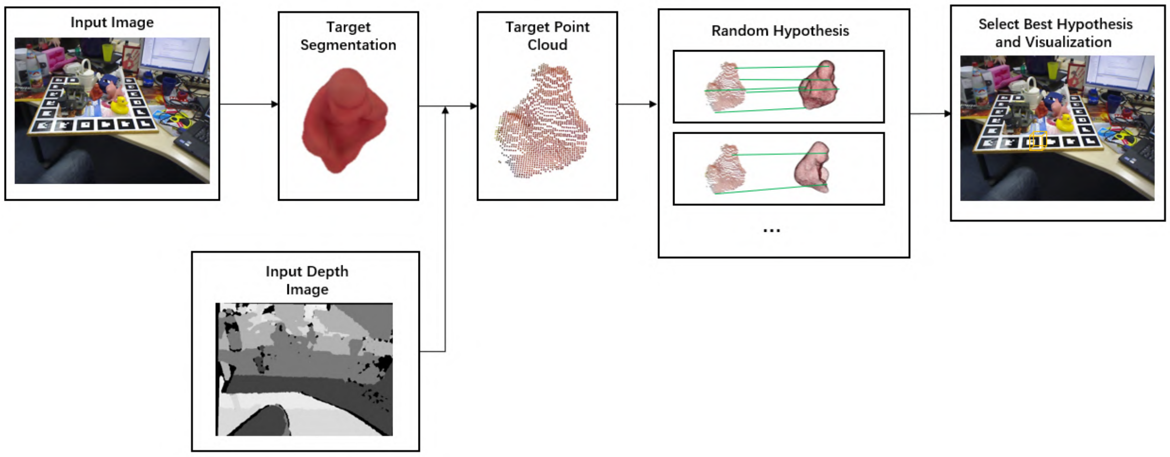
Template-based methods

The transitional 3D template-based methods are similar to the 2D approach, compared between the target and template library and choice best matching as Figure 2.11 (a) demonstrated. For the learning-based methods, the pioneering work is PointNet [24, 25] directly processes raw unstructured point clouds without transforming them to other types. PointNets only achieved classification and segmentation for a single object. Qi et al. [103] extended PointNet to propose Frustum PointNets, which leverages both mature 2D object detection algorithm and advanced 3D deep learning to implement 3D object detection in the context of autonomous driving.

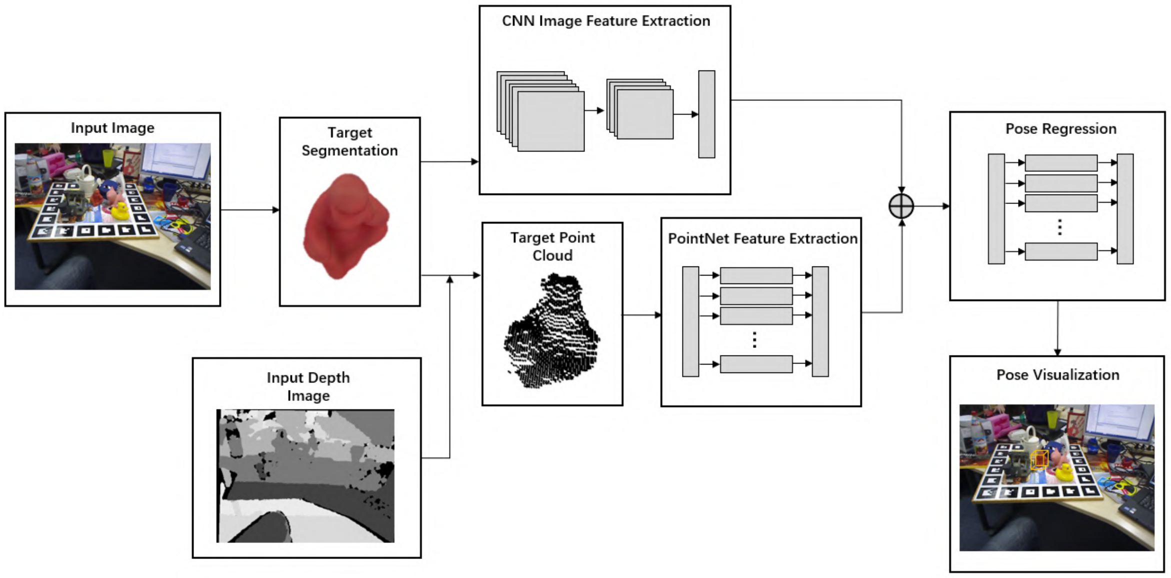
Inspired by the success of Frustum PointNets [103] and PointFusion [104], DenseFusion [15] was proposed to predict the 6D pose of a set of known objects. It fully embeds and fuses color information of RGB image and geometric information of point clouds at a per-pixel level to yield a pose compute for each pixel, and then generates the final 6D pose. Figure 2.11 (b) presents the overall DenseFusion architecture. Similar to DenseFusion [15] using PointNets [103] to extra 3D features, Gao et al. [105] proposed a 6D pose estimation network, CloudPose, which only relies on point cloud geometric information. Moveover, CloudPose [105] regresses translation and rotation separately compared with DenseFusion [15]. Subsequently, Gao et al. [117] utilize an augmented autoencoder to encodes 6D object pose evidence for pose estimation.

Voting-based methods

Voting-based methods commonly can be seen as an extension of correspondence-based approaches. At early stages, Tombari and Di Stefano [18] proposed an efficiency 6D



(a) Transitional 3D template-based method.



(b) Deep learning-based method [15].

Figure 2.11: Overview architecture of 3D template-based methods. The object image is from LineMOD dataset [8].

object pose detection algorithm by hough voting in a cluttered and occluded scene. Each feature point is linked to the model's centroid, allowing each scene feature to vote in a 3D Hough space to gather evidence for possible centroid positions in the current scene. And then Woodford et al. [106] improved 3D hough voting by the intrinsic and minimum-entropy Hough transform.

More recently, with the growth of 3D deep learning techniques, voting-based methods have been introduced into this task yielded encouraging results. Hua et al. [118] proposed a 6D pose estimation framework by employing a 3D point-to-keypoint voting strategy that uses a dense fusion feature to vote for the keypoints. What's more, Yoloff [107] utilize a regression CNN detecting a set of vectors representing the 3D keypoints of object. Those votes are then summed in a non-parametric method to get a reliable estimate of the keypoints. Although these methods are robust to the cluttered and occluded environment, the output space is enormous.

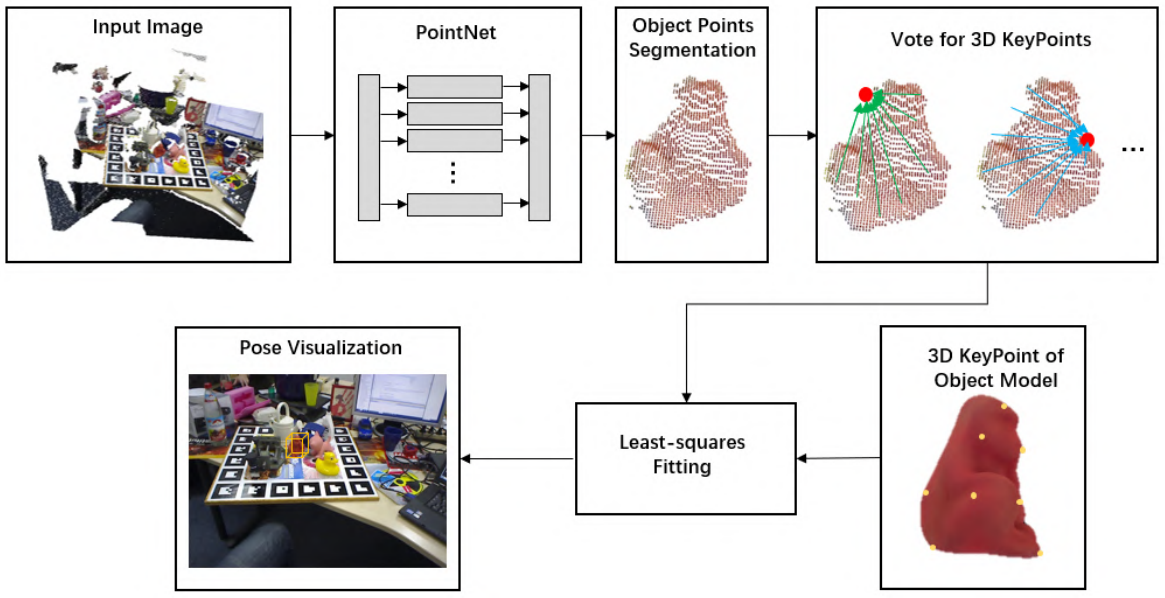


Figure 2.12: Overall all architecture of a 3D voting-based method, PVN3D He et al. [9]. The object image is from LineMOD dataset [8].

To tackle this challenge, He et al. [9] proposed PVN3D, that is, an extension of PV-Net [23] in the 3D domain shown in the Figure 2.12. PVN3D is composed by a

3D keypoint detection module, a semantic segmentation module and a center voting module. Keypoint detection module detects 3D keypoints by using a deep 3D Hough voting network, which learns to predict the point-wise 3D offset then vote for the 3D target keypoints. A least-squares fitting calculates the final 6D pose of the object. Additional, by combining vote features, Qi et al. [108] presented VoteNet, which can vote for the virtual centroid of objects from point clouds directly. However, for a partially obscured object, the virtual centre point prediction is inaccurate. By extending VoteNet [108], Qi et al. [109] proposed ImVoteNet, which use a hybrid 2D-3D voting scheme to covert the 2D votes to "pseudo" 3D votes. To ensure full use of both 2D and 3D features, ImVoteNet is a multi-towered architecture for training with gradient blending.

2.3.3 6D Pose Estimation for Particular Objects

Above sections review most generic approach for 6D pose estimation. However, there are two kinds of objects worth of advanced discussion, which are transparent objects (such as glass) and non-rigid objects (such as human, pants and socks).

6D Pose Estimation for Transparent Objects

The biggest challenge for transparent objects pose estimation is that depth sensor fails when objects without opaque, lambertian surfaces Liu et al. [119]. To overcome it, Liu et al. [119] proposed KeyPose, which is the first approach of correspondence-based pose estimation for transparent objects from stereo RGB images. KeyPose first crop a fixed-size rectangle from the left image, and a corresponding rectangle at same position from the right image. Then KeyPose utilize a CNN network to output disparity implicitly. Finally, KeyPose convert UVD value to 3D key points by reprojecting pixel errors. Similarly, [120] propose GhostPose, which utilize 3D keypoints and multi-view geometry to recover transparent object without object CAD model. In addition, many high quality dataset for transparent objects 6D pose estimation are created to improve the performance of detection, such as StereOBJ-1M[121], PhoCaL[122] and Keypose[119] etc. Overall, 6D pose estimation for transparent objects heavily rely on RGB image to predict 3D keypoint. Consequently, prediction results are extremely sensitive the quality of RGB image .

6D Pose Estimation for Non-rigid Objects

Apart from transparent objects, 6D pose estimation for non-rigid objects is another challenge. 6D pose estimation for non-rigid objects require the more global information extracted from larger receptive fields and are inescapably more sensitive to occlusions. Up to now, far too little attention has been paid to non-rigid objects pose estimation. Ge and Fan [123] proposed a pose estimation framework for human body using RGB-D data. This is a hybrid registration approach that use segment-aware articulated iterative closest point (SAICP) adapted from articulated iterative closest point (AICP).

For the non-rigid objects without fixed shape, it is a research area that had not been widely addressed [124]. However, the robotic manipulation for non-rigid objects has received important contributions in recent years. Wang et al. [125] present 3d-physnet, which can learn object body deformations of non-rigid objects. In addition, Nadon and Payeur [126] developed a manipulation system that can select an optimal grasp for controlling the shape of a non-rigid object. In summarise, the research on 3D non-rigid objects pose estimation is still in its infancy.

2.4 Summary

This chapter began by presenting relative preliminaries to 3D vision system. We introduce the commonly used 3D object representation types as well as 3D data acquisition methods. This knowledge provides guidance for the sensor and data selection. Afterwards, the important concept of object 6D pose, is introduced. In particular, we explain three different rotation representations and their mathematical conversion.

After that, the literature review concentrates on the relevant 3D environment reconstruction and objects' 6D pose estimation aspects. We discuss this from both a 2D and 3D perspective. After a technical review of the state-of-art, we identified a number of issues:

- With the current 3D environment reconstruction approaches, no matter whether 2D-based [46, 5] or 3D-based [55, 6, 59, 11], the performance of dense reconstruction for the small-scales texture-less objects does not meet expectations.
- For the RGB-based objects 6D pose estimation technology [14, 127, 97, 128], has low detection accuracy in cluttered and occluded scenes [9].
- The RGB-D-based 6D pose estimation methods [22, 4, 104, 103] improve the detection accuracy a lot, but they exploit the RGB information and depth channel separately, which causes features to be missing.
- The methods mentioned before are only evaluated using a standard dataset and often lack application in practical scenarios.

To bridge the above practical gaps, in the following chapters we propose a 3D scene reconstruction method using coloured points as an input, a learning-based 6D pose estimation system using a single RGB image as an input, and a point cloud-based 6D pose estimation approach. Finally, we develop a 3D intelligent system by integrating those novel techniques and applying this system in a nuclear waste disposal experiment.

Chapter 3

3D Scene Reconstruction from Point Cloud

3.1 Introduction

Modelling of the surrounding scene is fundamental for most robotic tasks, like rescue, guidance, and pick-and-place. An accurate model of the robot work space has been considered to be a prerequisite for an autonomous robot [129]. In section 2.2, we have reviewed different methods for 3D scene reconstruction. Image-based approaches severely rely on scene texture, while learning-based methods perform poorly in previously unseen objects. This motivated our use of point cloud data as an input for scene modelling as a point cloud provides both geometric and photo-metric information.

Point-cloud based environment reconstruction usually relies 3D point cloud registration as the most important part of its pipeline. To improve the accuracy and speed of point cloud registration, researchers have undertaken a large number of attempts [130, 131, 10, 132] to improve registration performance. In particular, ICP [57] and its variants [133, 134, 10] have become dominant approaches for point cloud registration. The traditional ICP algorithm are however affected by initial position and search methods. For highly overlapping and rich geometry of point cloud sets, the ICP approach perform quite well. However, this strategy suffers from limitations in scenes with little 3D structure because it severely relies on shapes and initial positions. Fortunately, some of recent works exploit colour information [134, 10] and powerful non-linear optimization [132, 135] to implement faster convergence speed and high alignment accuracy. Despite this, current environment reconstruction technologies cannot meet industrial performance requirements, especially for the small-scale texture-less scenes. Industrial objects are generally texture-less and irregular in shape. Most of the widely used reconstruction approaches are either noisy [11, 10] or completely miss [136] the fused output.

To bridge such contradictory technical challenges, we propose a novel point cloud-based reconstruction pipeline. This reconstruction system contains two parts: a). a point cloud registration system that computer the relation between two point cloud; b). a point cloud fusion strategy than can merge registered point cloud set into a whole environment model.

For the point cloud registration system, our method considers the colour information and geometric information and defines a joint optimization objective in three-dimensions. Moreover, we utilize a Levenberg-Marquardt method to replace the Gussion-Newton approach to optimize a nonlinear least-squares objective in ICP processing, which can improve accurate of registration.

As for point cloud fusion system, we develop a hierarchical coarse-to-fine strategy to overcome local minima and position drifting problems. Specifically, the input point cloud streams are divided into short fragments. On the first hierarchy level, we execute coarse-to-fine processing within sequences of a fragment. On the second hierarchy level, the new fragment is globally aligned against the previous fragment.

In summary, the main contributions are as follows:

- Inspired by colored-ICP [10], an our algorithm is proposed that joins the application of photo-metric and geometric terms thereby achieving accurate alignment. In addition, we consider employ Levenberg-Marquardt method to process the minimize objective function. Therefore, our point cloud registration method as known as LM-ICP.

- Development of a hierarchical coarse-to-fine point cloud fusion strategy that can generate a high-quality model of the environment.

The proposed system is validated on a prevailing data set [7, 12] and real-world scenarios. The experiments show that the system is robust, rapid and precise in comparison to some some widely used systems [10].

In the rest of this chapter, details of the method in Section 3.2, then the evaluation results are described in Section 3.3, and conclusions are drawn in Section 3.4.

3.2 Point Cloud Fusion System

For ease of presentation, we denote the ith point cloud by $\boldsymbol{P}_i$, and the corresponding camera pose by $\boldsymbol{R}_i$ so that $\boldsymbol{T}_i(\boldsymbol{P}_i) = \boldsymbol{R}_i\boldsymbol{P}_i + \boldsymbol{t}_i$ is the relative transformation of the point cloud from the camera frame to the world frame. Here $\boldsymbol{t}_i$ is the camera location in the world frame and $\boldsymbol{R}_i$ is the rotation that brings the camera frame aligned with the world frame.

Given a point cloud set $\boldsymbol{S} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_n\}$, the aim is to obtain a best set of camera transforms $\{\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_n\}$ so that all point clouds align well. First set the first point cloud as the world coordinate system, so the reconstruction model $\boldsymbol{M}$ is:

$$\boldsymbol{M} = P_1 \cup \bigcup_{i=2}^{n} \boldsymbol{T}_i(\boldsymbol{P}_i) \tag{3.1}$$

For the computation of transformation $\boldsymbol{T}_i$, the recursive ICP algorithm mainly involves two steps:

1. Find the correspondence set $\boldsymbol{\kappa} = \{(\boldsymbol{p}, \boldsymbol{q})\}$ between the already merged point cloud $\boldsymbol{P}_{i-1}$ and the new point cloud $\boldsymbol{P}_i$ for a given a transformation $\boldsymbol{T} = [\boldsymbol{R}, \boldsymbol{t}]$ estimated

by inertial sensors placed on the camera. This requires the minimisation of

$$C(\kappa) = \sum_{\kappa=(p,q)} \|p - \boldsymbol{R}q - \boldsymbol{t}\|^2$$

2. Minimize an error cost function $F(\boldsymbol{T})$ to update the new transformation $\boldsymbol{T}_i$ to a more precise one. The most common error cost function for point-to-point ICP is:

$$F(\boldsymbol{T}) = \sum_{(R,t)} \sum_{\kappa=(p,q)} \|p - \boldsymbol{R}q - \boldsymbol{t}\|^2 \tag{3.2}$$

In the rest the short notation $\boldsymbol{T}q = \boldsymbol{R}q + \boldsymbol{t}$ will be used where $\boldsymbol{T} = [\boldsymbol{R}, \boldsymbol{t}]$.

3.2.1 Coloured Point Cloud Registration

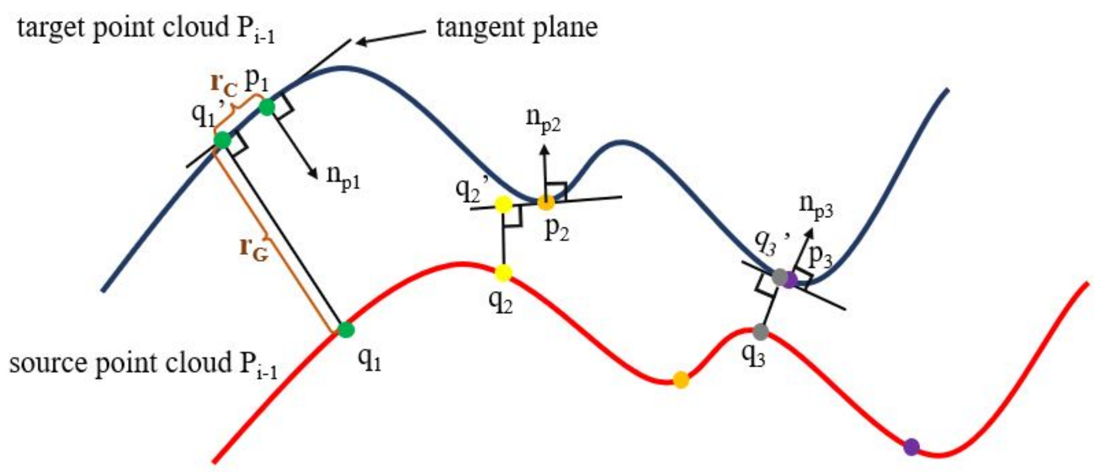


Figure 3.1: The geometric and photometric error between two point clouds

In [10], the authors pioneered a joint optimization objective that converts both photometric and geometric terms as a continuous function:

$$F(T) = \sigma F_G(T) + (1 - \sigma) F_C(T) \tag{3.3}$$

where F_G and F_C are the nonlinear least-squares function for geometric and photometric items, respectively and $\sigma \in [0, 1]$ is the weight of those two items.

As Figure 3.1 illustrates, r_G is the error distance between tangent of p and q, so F_G is actually a point-to-plane ICP algorithm:

$$F_G(\boldsymbol{T}) = \sum_{(p,q)\in\kappa} ((p - \boldsymbol{T}q) \cdot n_p)^2 \tag{3.4}$$

where n_p is the normal of point p, $\kappa = (p, q)$ is the correspondence set from target point cloud $\boldsymbol{P}$ and source point cloud $\boldsymbol{Q}$, and $q^{'}$ is the projection of point q onto the tangent of p. The point-to-plane ICP only rely on geometric distance to align the two point cloud, so it's not reliable in some situation. As Figure 3.1 shown, different colours represent different point. The geometric error r_G of (q_2, p_2) and (q_3, p_3) is small but (q_2, p_2) and (q_3, p_3) are wrong correspondence points. Therefore, we use the photometer error r_c as a supplement to distinguish those points.

The photometric term F_C represents a continuous and differentiable colour function:

$$F_C(T) = \sum_{(p,q)\in\kappa} (C_p(q') - C(q))^2 \tag{3.5}$$

where $C(p)$ is the colour intensity of each point p and $C_p()$ represent a continuous colour defined on the tangent plane of point p function:

$$C(p) = 0.299 * R(p) + 0.587 * G(p) + 0.114 * B(p) \tag{3.6}$$

where $R(), G(), B()$ are point intensity for red, green and blue respectively. Compared with an original method utilizing average intensity value, grayscale can avoid many ambiguities. For example, the average intensity of red (255,0,0) and green (0,255,0) are the same, while grayscale can distinguish them easily. The function $C_p(q')$ approximately equals, to its first-order approximation:

$$C_p(q') \approx C(p) + \boldsymbol{d_p} q' \tag{3.7}$$

We use grayscale conversion eqn. (3.6) representing the intensity value of each point to calculate the colour gradient.

$$q' = q - n_p(q - p)^T n_p \tag{3.8}$$

$\boldsymbol{d_p}$ is the gradient of C_p and can be calculated by utilizing least-squares fitting to $\{\bar{p} \in N_p\}$, and $\bar{p}$ is point $\boldsymbol{p}$'s nearest neighborhood.

$$\begin{aligned} L(\boldsymbol{d_p}) &= \sum_{\bar{p} \in N_p} (C_p(\bar{p}^{'} - p) - C(\bar{p}))^2 \\ &\approx \sum_{\bar{p} \in N_p} (C(p) + \boldsymbol{d_p}^T(\bar{p}^{'} - p) - C(\bar{p}))^2 \end{aligned} \tag{3.9}$$

Despite achieving a good registration result, this algorithm still has many drawbacks affecting the quality of the alignment. Therefore, we consider the following improvements for this algorithm to improve the accuracy and robustness of the registration.

In addition, for objective optimization, we propose a Levenberg-Marquardt Iterative Closest Point (LM-ICP). Specifically, LM-ICP begins with an initial transformation $\boldsymbol{T_i^0}$ and executes the optimization iteratively. In each iteration, we calculate the residual $\boldsymbol{r}$ and Jacobian $\boldsymbol{J_r}$ at $\boldsymbol{T_i^{k-1}}$ as estimated in the last iteration, and solve the Levenberg–Marquardt function A.10 to obtain increment $\xi = (t_x, t_y, t_z, \alpha, \beta, \gamma)$. The reason we use LM function to replace the Gauss-Newton function is that the LM algorithm has more tolerance of inaccuracy for initial approximation [137]. We expect that the LM method can prove a robust optimization processing for texture scene alignment.

$$(\boldsymbol{J_r^T}\boldsymbol{J_r} + \mu \boldsymbol{I})\xi = -\boldsymbol{J_r^T}\boldsymbol{r} \tag{3.10}$$

where damping parameter μ is adjusted at each iteration, which influences both the magnitude and direction of the step, I is a 6×6 identity matrix and $\boldsymbol{r}$ is the residual of objective function eqn. (3.3):

$$\boldsymbol{r} = \sqrt{(1-\sigma)F_C(\boldsymbol{T_i})} + \sqrt{\sigma F_G(\boldsymbol{T_i})} \tag{3.11}$$

To compute the Jacobian, we need the partial derivatives of the residuals, which denotes as $\nabla \boldsymbol{r}$. It is:

$$\begin{aligned}\boldsymbol{J_r} &= \nabla \boldsymbol{r} \\ &= \sqrt{(1-\sigma)}\nabla r_C(\boldsymbol{T_i}) + \sqrt{\sigma}\nabla r_G(\boldsymbol{T_i}) \\ &= \sqrt{(1-\sigma)}\boldsymbol{d_p}\boldsymbol{J_q} + \sqrt{\sigma}\boldsymbol{n_p}^T\end{aligned} \tag{3.12}$$

where $\boldsymbol{J_q}$ is the Jacobian matrix computed from eqn. (A.4). According to the knowledge introduced in section A.2, parameter μ is controlled by gain ratio φ:

$$\varphi = \frac{F(\boldsymbol{T}) - F(\boldsymbol{T} + \boldsymbol{\tau}(\xi))}{|L(\mathbf{0}) - L(\boldsymbol{\xi})|} \tag{3.13}$$

Note that $F(\boldsymbol{T}) - F(\boldsymbol{T} + \boldsymbol{\tau}(\xi))$ represents the root-mean-square deviation (rmsd) between the previous target and transformed target. The denominator is obtained from eqn. (A.13), so:

$$L(\mathbf{0}) - L(\boldsymbol{\xi}) = -\boldsymbol{\xi}^T\boldsymbol{J_r}^T\boldsymbol{r} - \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{J_r}^T\boldsymbol{J_r}\boldsymbol{\xi} \tag{3.14}$$

that is, the ratio of the actual and estimated decrease in residuals.

Putting it all together, the gain ratio φ from eqn. (A.14) can be written as:

$$\varphi = \frac{RSME(\boldsymbol{T}, \boldsymbol{\xi})}{|-\boldsymbol{\xi}^T\boldsymbol{J_r}^T\boldsymbol{r} - \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{J_r}^T\boldsymbol{J_r}\boldsymbol{\xi}|} \tag{3.15}$$

A small gain ratio φ means that we should increase the φ value and improve the penalty on large steps. While a big φ value suggests that the actual and estimated values are a good approximation for the computed ξ, and the φ value should be reduced. Solving eqn. (3.10), we can obtain increment parameter ξ, which contains rotation and translation components. Then, we utilize following equation converting the 6-vector ξ to a 4×4 translation matrix $\boldsymbol{T_i}$:

$$\boldsymbol{T_i^k} \approx \begin{pmatrix} 1 & -\gamma & \beta & t_x \\ \gamma & 1 & -\alpha & t_y \\ -\beta & \alpha & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{T_i^{k-1}} \tag{3.16}$$

Algorithm 1 Levenberg-Marquardt based point cloud alignment

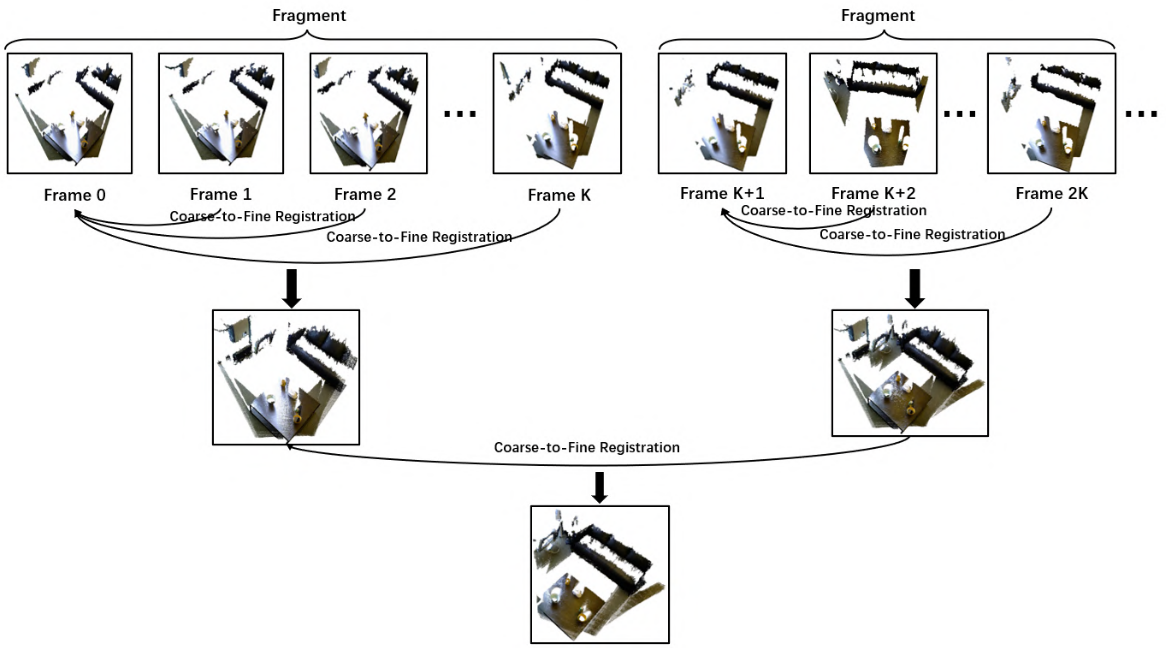
```
Input: Target point cloud $\boldsymbol{P}_{i-1}$; Source point cloud $\boldsymbol{P}_i$; Initial transformation $\boldsymbol{T}_i^0$;
Output: Transformation $\boldsymbol{T}_i$ aligns $\boldsymbol{P}_i$ to $\boldsymbol{P}_{i-1}$
 1: for $\boldsymbol{p} \in \boldsymbol{P}_{i-1}$ do
 2:     Compute $\boldsymbol{d_p}$ through miniming eqn. (3.9)
 3:     Initializing $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, $\boldsymbol{\mu} = \boldsymbol{\nu} = 2$
 4: Compute the correspondence set $\boldsymbol{\kappa} = \{(\boldsymbol{p}, \boldsymbol{q})\}$, $\boldsymbol{p} \in \boldsymbol{P}_{i-1}$ and $\boldsymbol{q} \in \boldsymbol{P}_i$
 5: while less than maximum iteration do
 6:     $\boldsymbol{r} \leftarrow \boldsymbol{0}, \boldsymbol{J_r} \leftarrow \boldsymbol{0}$
 7:     for $(\boldsymbol{p}, \boldsymbol{q}) \in \phi$ do
 8:         Compute residual $\mathbf{r}$ and Jacobian $\boldsymbol{J_r}$ at $\boldsymbol{T_i}$.
 9:     Compute $\boldsymbol{\mu}$ according to eqn. (3.4)
10:     Solve Levenberg-Marquardt function to obtain $\boldsymbol{\xi}$
11:     Update $\boldsymbol{T}_i$ unitizing eqn. (3.16)
12:     if converged do
13:         return $\boldsymbol{T}_i$
```

In the next iteration, we recompute $\boldsymbol{T_i}$ and repeat until convergence. This point cloud registration strategy is summarized in Algorithm 1. Our code is available at https://github.com/50618861/reconstruction_algorithm_code.

3.2.2 Hierarchical Coarse-to-fine Pipeline

With scale the incremental method suffers from the accumulation of pose drift [138]. To avoid geometric drift, we exploit a hierarchical fusion strategy. The input point cloud streams are divided into short fragments as Figure 3.2 shown. On the first hierarchy level, we execute coarse-to-fine processing within sequences of a fragment. On the second hierarchy level, the new fragment is globally aligned against the previous fragment.

Firstly, we make an initial fragment by aligning five consecutive frames in the input



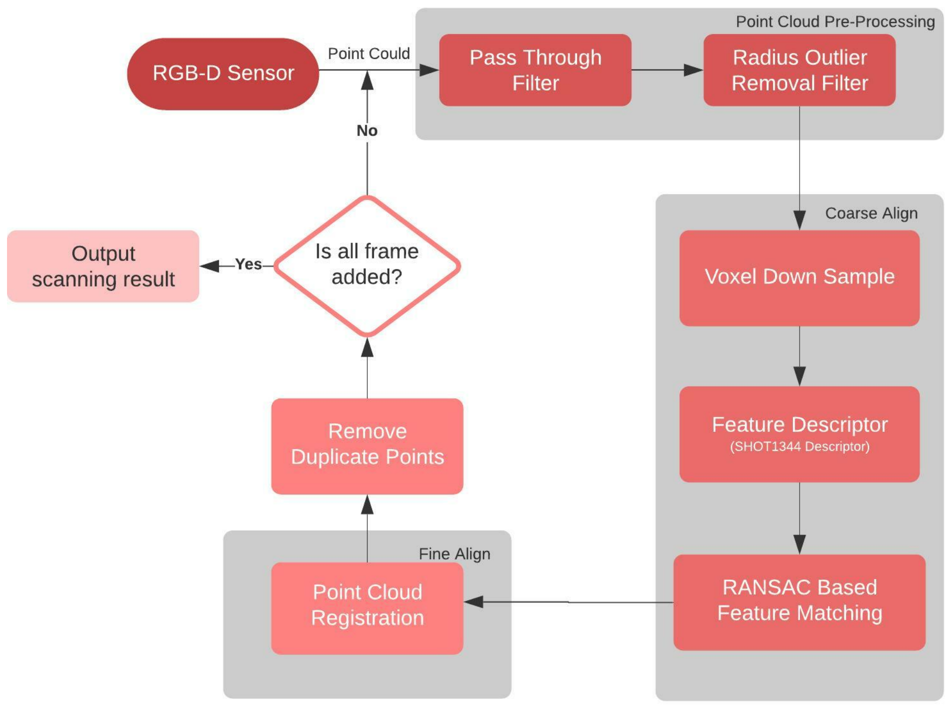
Fragment
Fragment
Frame 0
Frame 1
Frame 2
Frame K
Frame K+1
Frame K+2
Frame 2K
Coarse-to-Fine Registration
Coarse-to-Fine Registration
Coarse-to-Fine Registration
Coarse-to-Fine Registration
Coarse-to-Fine Registration

Figure 3.2: Illustration of Hierarchical Coarse-to-fine Pipeline

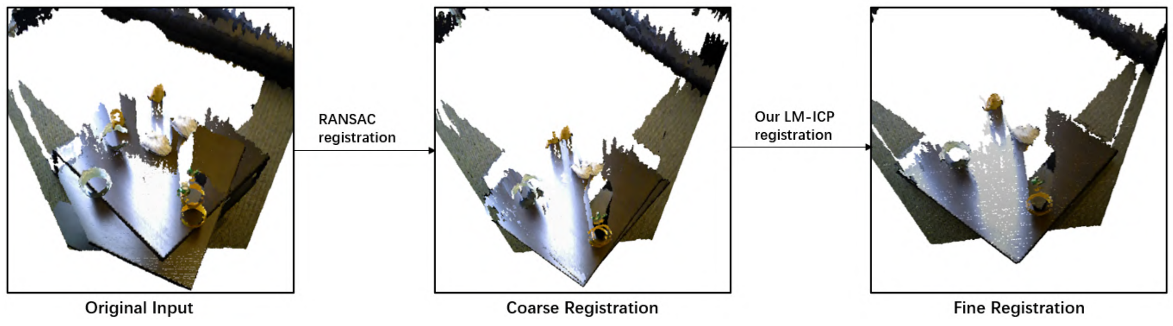
point cloud stream. Subsequently, every five consecutive point clouds will be made into a fragment. Since adjacent point clouds have extensive overlap, and the pose variation within a fragment is minuscule, we set each first frame to the identity matrix. Once a fragment is finished, it will be aligned with the previous fragment already registered to the global map, and then the transformed fragment will be merged into the global map. To guarantee that the point cloud pose after convergence is accurate, we apply the our LM-ICP method shown in Figure 3.3 to each alignment, including local and global point cloud registration.

Figure 3.3 illustrates the coarse-to-fine registration, which can ensure a precise fusion result. The original raw point cloud data from 3D sensors are noisy and cluttered owing to the camera's field of view, measurement errors and light reflections; this complicates the point cloud processing. Consequently, before the feature extraction, we need to remove the redundant and outlier points.

First, the raw point cloud is filtered by a pass-through filter that can delete the points outside the workspace. Secondly, the filtered points will pass to a radius outlier



(a) Block diagram of point cloud fusion system

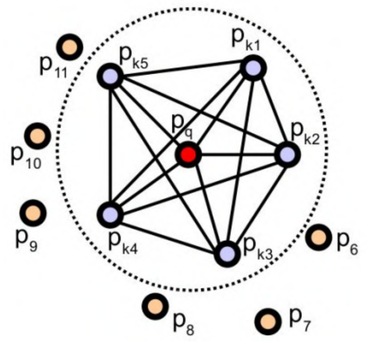


(b) Illustration of coarse-to-fine processing

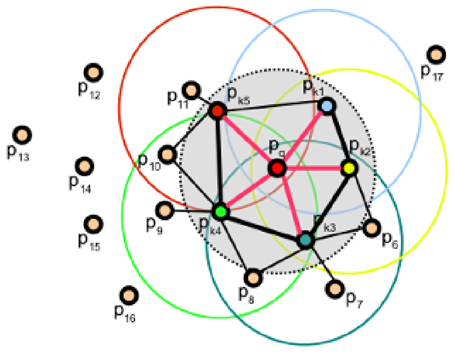
Figure 3.3: Point cloud fusion system

removal filter that can clear outlier points according to the number of neighbours. The point will be seen as an outlier if the number of neighbours within a given radius is less than the threshold.

The ICP based method is sensitive to an initial position of point cloud. To avoid getting stuck in local minima, we utilize a coarse global alignment as initialization. In the first place, the denoised point cloud will be downsampled by an approximate voxel filter, which can speed up the alignment process while retaining features detail. Next, we will perform a RANSAC registration [11] to obtain a rough translation between source and target point cloud.



(a) The influence region diagram for a k-neighborhood set (blue) centered at p_q (red).



(b) The influence region diagram for FPFH. The query point p_q (red) is only connected between itself and its neighbors (illustrated using red lines). Black lines indicate extra connection and thicker lines means counted twice

Figure 3.4: Point Fearure Histogram (PFH) and Fast Point Fearure Histogram (FPFH) descriptor

In each iteration of the RANSAC registration, it randomly selects some points from the source $\boldsymbol{P_{i-1}}$. The corresponding points of selected points in the target $\boldsymbol{P_i}$ are found by matching a 3-dimensional FPFH feature [67] between source and target point cloud.

Note that Fast Point Feature Histogram (FPFH) [67] simplifies the Point Feature Histogram (PFH) descriptor [139], which significantly minimises the time it takes to compute. Figure 3.4(a) demonstrate the PFH descriptor of point $\boldsymbol{p_q}$. The point $\boldsymbol{p_q}$ and its $\boldsymbol{k}$ neighbors with distance less than the radius $\boldsymbol{r}$ are fully interconnected. The dimensional of a PFH feature vector is 125 (5^3), and it has a complexity of O(k^2). As for the FPFH descriptor, it considers only the connection between the point $\boldsymbol{p_q}$ and its k neighbours, eliminating any extra connection between neighbours. Next, we repeat this process for all point $p \in P_i$, and the FPFH of point $\boldsymbol{p_q}$ will merge with its neighbours', weighted based on the distance, to compose the eventual descriptor. The final FPFH descriptor only uses 11 binning subdivisions that output a 33-byte (3×11) array of float values and simplifies the computational complexity to O(k).

After identifying the correspondence between source and target point cloud, we perform the RANSAC iterations provided by [11] for correspondence points to obtain a coarse alignment result that is employed as initialisation of the ICP processing. Following a rough global registration, we use the point cloud registration algorithm Algorithm 1 to refine the alignment further. In the hierarchical strategy mentioned previously, this coarse-to-fine pipeline is performed for each alignment. Algorithm 2 summarize the implementation of the point cloud fusion.

Algorithm 2 Point cloud fusion system

```
Input: Point cloud sequence $\boldsymbol{P}$;
Output: Environment model $\boldsymbol{M}$;
 1: Set the number of point cloud $\boldsymbol{k}$ in a fragment (default $k = 10$)
 2: Initialize fragment count $\boldsymbol{n} = 1$
 3: while new point cloud input do
 4:     for point cloud $\boldsymbol{P} \in \boldsymbol{P}_{i,i=n*(1,2,3,\ldots,k)}$ do
 5:         Pre-process for $\boldsymbol{P}$ to remove noise
 6:         Extract feature descriptor for $\boldsymbol{P}$
 7:         Use RANSAC registration to obtain a rough translation
 8:         Employ Algorithm 1 to refine the translation
 9:         Generate the point cloud fragment.
10:         $n = n + 1$.
11:     if $n == 1$ do
12:         Set first point cloud fragment as globe map $\boldsymbol{M}$
13:     else
14:         Registered point cloud fragment to globe map $\boldsymbol{M}$
15: Output the environments point cloud model $\boldsymbol{M}$
```

3.3 Experiments

3.3.1 Datasets

The proposed point cloud fusion algorithm Algorithm 2 is evaluated on two widely used datasets: the RGB-D Scenes Dataset v.2 and the SUN3D Dataset.

- RGB-D Scenes Dataset [7]. The RGB-D Scenes Dataset v2 comprise 14 scenes including furniture (chair, coffee table, sofa, table) as well as a subset of the objects in the RGB-D Object Dataset (bowls, caps, cereal boxes, coffee mugs, and soda cans). We randomly select six scenarioes for evaluation experiment, they are rgbd-scenes-v2-scene_01, rgbd-scenes-v2-scene_02, rgbd-scenes-v2-scene_04, rgbd-scenes-v2-scene_07, rgbd-scenes-v2-scene_10 and rgbd-scenes-v2-scene_13.
- SUN3D Dataset [12]. SUN3D dataset is a place-centric dataset. It contains 415 sequences collected for 254 different areas in 41 distinct buildings. Evaluations were performed on two scenes from the the SUN3D dataset: sun3d-harvard_c6-hv_c6_1 and sun3d-harvard_c11-hv_c11_2.

3.3.2 Metrics

To evaluate the performance of point cloud registration and reconstruction, we employ a widely used metric. This metric is absolute pose error (APE), which measures the error in the absolute motion between calculated pose and ground truth. We define the APE matrix at frame i as:

$$Error_i = \boldsymbol{Q_i}^{-1}\boldsymbol{P}_i^{-1} \tag{3.17}$$

where Q_i denotes ground truth pose and P_i means estimate pose. The APE is generally separated into two parts: translation and rotation. We use the root mean square error(RMSE) indicating APE error:

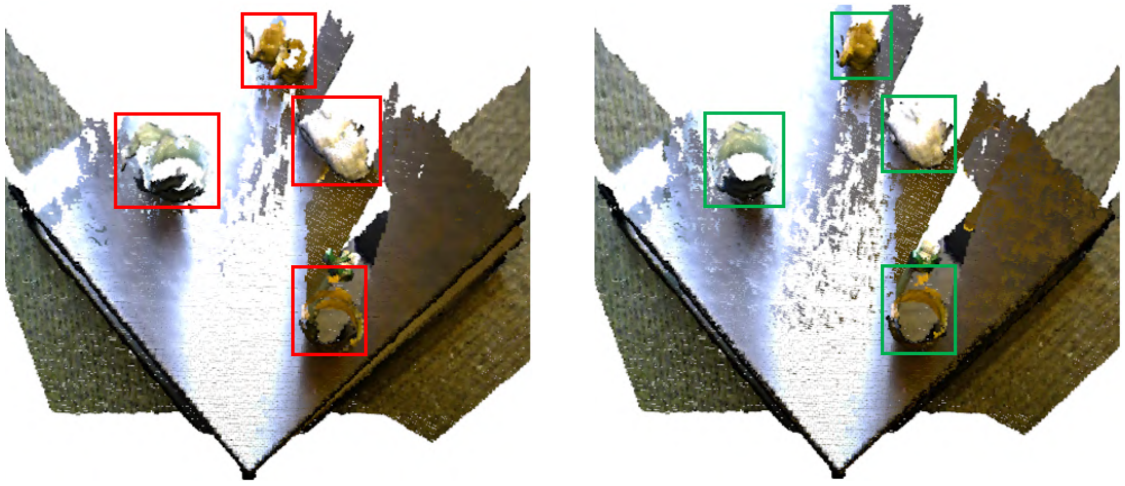
$$APE^i_{trans} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \parallel (trans(Error_i) \parallel^2)} \tag{3.18}$$

As for rotation representation we use following equation:

$$APE^i_{rot} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \angle \parallel (rot(Error_i) \parallel^2)} \tag{3.19}$$

Apart from RMSE, we also use standard deviation (STD) as indices.

3.3.3 Evaluation On RGB-D Scenes Dataset v2



(a) The registration result from coloured-ICP[10]

(b) The registration result using Algorithm 2

Figure 3.5: Comparison of proposed approach with coloured-ICP [10] in RGB-D Scenes dataset [7]. Red boxes indicate the misaligned part and green boxes show the correctly aligned part

First two point clouds are selected from rgbd-scenes-v2-scene_01 dataset to conduct evaluation for point cloud registration. From Figure 3.5 (a), we can notice that the environment model registered by [10] suffers from misalignment and drifting. In contrast, proposed method accurately aligns the small-scale textureless model and keeps geometric information as much as possible. Such high-quality results are beneficial for remote teleportation, point cloud-based deep learning, etc.

Following the above, quantitative evaluation is conducted to make comparisons with different registration algorithms [140, 141, 10, 11] using metrics we described before. Table 3.1 to 3.6 present the quantitative result tested on six different scenes from RGB-D Scenes Dataset V2. From those tables, we can notice that proposed algorithm Algorithm 2 method outperforms all the considered competitors in translation terms. But the rotation generated by competitors slightly surpass us in scene_02, scene_04, scene_10, scene_13. We argue the LM optimisation may cause this. The translation has an enormous impact on the objective function 3.3. To minimise the objective function, the optimisation sacrifices some rotational accuracy ensuring translational accuracy. It is particularly worth noting that all the competitors suffer from the drifting problem that causes significant errors in scene_07. In contrast, proposed method keep a good performance on it.

Additionally, the average running time is reported for one registration for the different techniques in Table 3.1. The running time is measured on a PC with an Intel Core i5-7300HQ CPU and Geforce GTX 1050 GPU. The point-to-plane ICP has the fastest processing speed because the source points only need to align with the tangent plane of targets. The coloured ICP adds the photometric terms on the basis of point-to-plane ICP, so its speed is slower than point-to-plane ICP. Proposed method adopts a complex LM optimisation (shown in Algorithm 1) and fusion strategies (shown in Algorithm 2); therefore, the speed of proposed method slightly lag in point-to-plane and coloured ICP. But it is still better than point-to-point ICP.

To highlight the performance of point cloud registration, comparison is also made with a widely-used multiway registration method [11]. The competitor's method [11] is an Open3D build-in global registration method exploiting pose graph optimization. A pose graph is composed of node and edge. A pose is attached to each node, and the neighbouring nodes are registered with ICP algorithm. Each node edges have to connect both adjacent and non-neighbouring nodes, which increases the complexity

Table 3.1: Quantitative evaluation on rgbd-scenes-v2-scene_01 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.3013	0.1786	13.7268	8.6452	0.6809
Point-to-plane ICP	0.1469	0.0672	**1.3465**	**0.1786**	**0.6809**
Coloured ICP	0.11979	0.06162	2.2195	0.72081	0.6959
Algorithm 2	**0.07550**	**0.03729**	4.9032	2.8109	0.6984

Table 3.2: Quantitative evaluation on rgbd-scenes-v2-scene_02 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.21446	0.08898	17.9308	5.0871	0.7309
Point-to-plane ICP	0.12686	0.04131	3.37871	2.40242	**0.7236**
Coloured ICP	0.07570	0.0332	2.0001	0.70899	0.77442
Algorithm 2	**0.02303**	**0.0171**	**0.797641**	**0.581682**	0.7814

Table 3.3: Quantitative evaluation on rgbd-scenes-v2-scene_04 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.3542	0.1656	18.9488	7.4194	**0.7777**
Point-to-plane ICP	0.1525	0.06423	5.98226	4.0013	0.84267
Coloured ICP	0.10712	0.050034	**3.60742**	**1.24233**	0.80054
Algorithm 2	**0.074848**	**0.05008**	4.8949	3.4885	0.8245

Table 3.4: Quantitative evaluation on rgbd-scenes-v2-scene_07 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.415462	0.16143	40.9519	22.3307	0.68704
Point-to-plane ICP	0.17255	0.0787	15.7292	10.6421	**0.67622**
Coloured ICP	0.222245	0.08087	17.5292	10.7206	0.69451
Algorithm 2	**0.0537**	**0.04173**	**2.15844**	**1.79306**	0.8014

Table 3.5: Quantitative evaluation on rgbd-scenes-v2-scene_10 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.1853	0.08189	11.3591	4.5891	0.97254
Point-to-plane ICP	0.083937	0.03939	**2.3737**	**0.8985**	**0.89207**
Coloured ICP	0.069128	0.032017	3.14501	1.3648	0.91484
Algorithm 2	**0.06360**	**0.0259**	2.6881	1.7036	0.9286

Table 3.6: Quantitative evaluation on rgbd-scenes-v2-scene_13 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	0.1019	0.05808	8.5058	4.9552	0.4509
Point-to-plane ICP	0.03101	0.01249	**0.6435**	**0.3317**	0.4468
Coloured ICP	0.0316	0.0143	0.9136	0.3829	**0.4277**
Algorithm 2	**0.0227**	**0.0159**	1.1101	0.7255	0.5014

of optimization. Figure 3.6 shows that Algorithm 2 is more close to the ground truth trajectory compared with a global registration method [11]. In figure 3.6, the black line is the ground truth trajectory, the blue line is trajectory obtained by our Algorithm 2 and red line represent competitor's result. It is clear that our method has smaller errors in five scenes. For scene rgbd-scenes-v2-scene_07, both methods have a good performance. As for the reconstruction speed, an 80 frame point cloud dataset generates 80 nodes and 3240 edges that consume 70 minutes to compute in the validation experiment. Although Choi et al. [11] proposes a novel pose graph optimization approach as post-processing for ICP algorithms, the reconstruction quality of proposed method is slightly superior without any post-processing refinement. Obviously, this global registration method is not ideal for large-scale point cloud data processing.

In addition, the qualitative reconstruction result is illustrated in Figure 3.7. For visualization, scalable TSDFs Volume integration from open3d is employed to create the mesh. Note that a TSDFs is a three-dimensional voxel array representing objects inside a volume of space, with each voxel labelled with the distance to the nearest

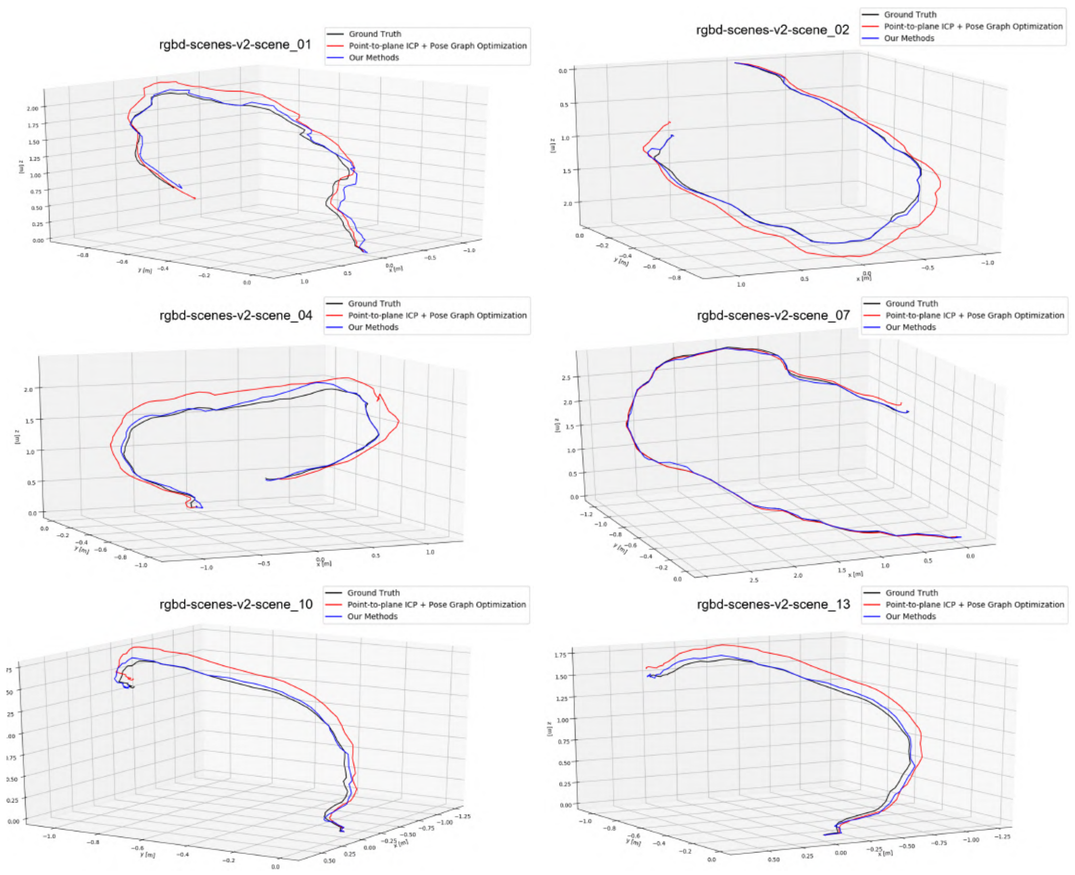
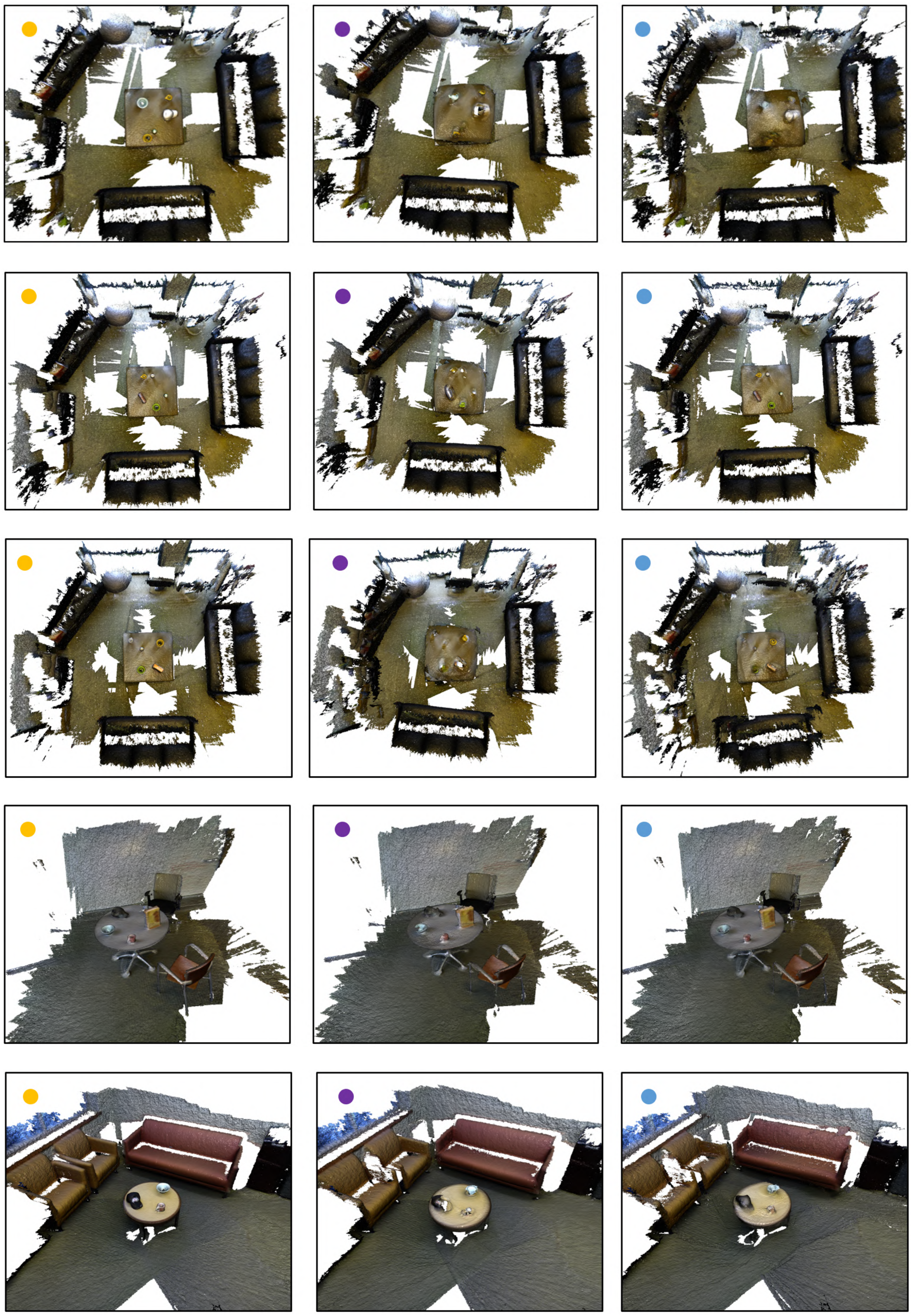


Figure 3.6: Trajectory comparison between ground truth, Choi [11] and proposed system

surface. In Figure 3.7, from top to bottom, they are reconstructed model of rgbd-scenes-v2-scene_01, rgbd-scenes-v2-scene_02, rgbd-scenes-v2-scene_04, rgbd-scenes-v2-scene_07, rgbd-scenes-v2-scene_10 and rgbd-scenes-v2-scene_13. For the scene_07 and scene_13, there is no big difference, and both proposed method and Choi et al. [11] approach are close to the ground truth. This also coincides with the quantitative data shown in Table 3.7 and previous tables. In the rest of scenes, [11]'s results have some obvious flaws, such as the black hat is partially missing in the scene_10. Despite the reconstruction model is not perfect, they recover the basic structure of the scene.

Overall, the proposed point cloud fusion Algorithm 2 is better performing in both qualitative and quantitative evaluation. Although proposed method is slightly slower



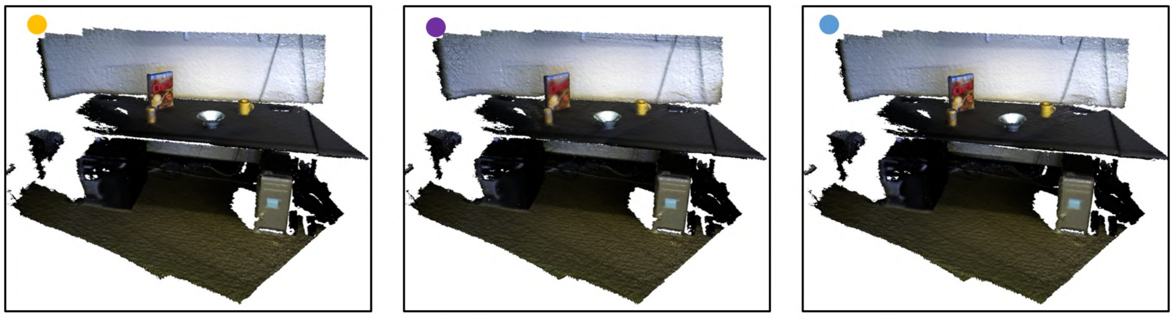


Figure 3.7: **Reconstruction of scenes from the RGB-D Scenes Dataset**. Yellow dot in top-left corner indicates ground truth, purple dot represents pose graph method [11], and blue dot is proposed approach

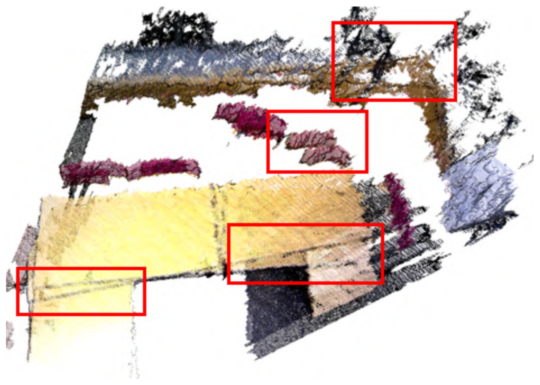
Table 3.7: Quantitative evaluation result using multiway registration method [11]

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)
scene_01	0.1408	0.0675	5.5360	3.0500
scene_02	0.1533	0.05751	6.0068	2.98532
scene_04	0.1971	0.08551	5.83367	2.794435
scene_07	0.071655	0.045679	3.0690	1.9492
scene_10	0.114607	0.0494	3.3562	1.5754
scene_13	0.0266	0.0070	0.6670	0.3394

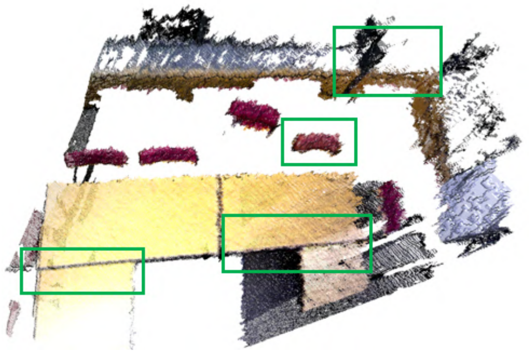
than other registration methods due to a complex optimization procedure, the discrepancy is negligible.

3.3.4 Evaluation On SUN3D Dataset

To further explore large-scale point cloud data performance, we conduct similar evaluation experiments on the SUN3D dataset. The quantitative evaluation is shown in Table 3.8 and 3.9, and the results are consistent with the previous; Algorithm 2 outperforms all the considered competitors in both translation and rotation terms, in spite of being slightly slower. The total running time increases because point cloud scenes from the



(a) The registration result from coloured-ICP[10]



(b) The registration result using Algorithm 2

Figure 3.8: Comparison of Algorithm 2 with coloured-ICP [10] in SUN3D dataset [12]. Red boxes indicate the misaligned part and green boxes show the correct aligned part

SUN3D dataset contain more points than the RGB-D Scenes dataset. Note that the error for both translation and rotation has a huge increase duo to the complexity and difficulty of large-scale datasets. It is worth to consider that how to improve it in the future research.

Table 3.8: Quantitative evaluation on the sun3d-harvard_c6-hv_c6_1 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	4.9943	**1.0957**	160.8450	5.3849	2.8679
Point-to-plane ICP	4.6494	2.3519	148.6320	10.0629	2.8127
Coloured ICP	5.9207	2.7083	**112.52**	23.7757	**2.7631**
Algorithm 2	**3.8964**	1.2877	130.80	**4.3496**	3.2583

For the qualitative evaluation, Figure 3.8 demonstrates registration results utilizing coloured-ICP [10] and Algorithm 2. It is obvious that proposed method solves unaligned problems and presents a high-quality result in both the table part and the cluttered top-right area. While for the reconstruction of the whole scene, both two methods have flaws. The pose graph method [11] is completely incorrect as shown in Figure 3.9 that the trajectory of the pose graph method is totally completely devi-

Table 3.9: Quantitative evaluation on the sun3d-harvard_c11-hv_c11_2 dataset

	Trans RMSE(m)	Trans STD(m)	Rot RMSE(°)	Rot STD(°)	Average Time(s)
Point-to-point ICP	2.7590	**0.7140**	169.3673	7.8662	2.3461
Point-to-plane ICP	4.4297	1.3323	153.5889	3.8824	**2.1310**
coloured ICP	2.3814	1.2323	108.5645	3.8824	2.1832
Algorithm 2	**1.2784**	1.1056	**71.4323**	**6.2985**	3.2583

ated from the ground truth. In addition, Figure 3.10 also show the reconstruction of pose graph method [11] are failed. In contrast, result generated by Algorithm 2 can recover the basic layout and furniture despite missing a corner, as illustrated in the bottom-right of Figure 3.10.

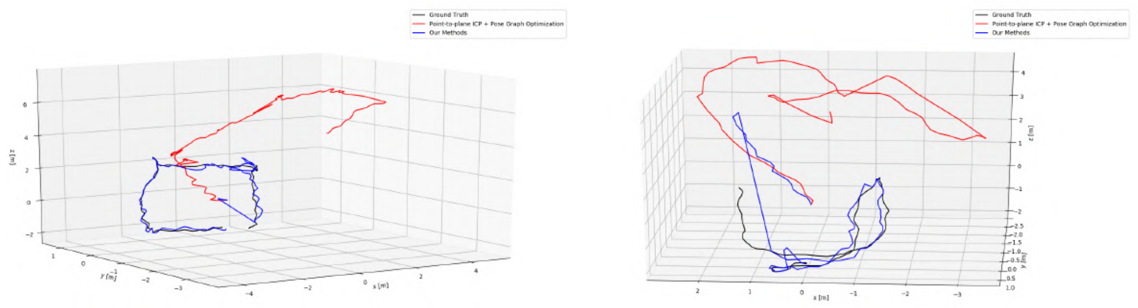


Figure 3.9: **Trajectory comparison between ground truth, Choi [11] and Algorithm 2.** Left: result on *sun3d-harvardc6-hv_c6_1* scene. Right: result on *sun3d-harvardc11-hv_c11_2* scene.

Overall, the reconstruction using a pure point cloud is more challenging on a large-scale dataset. Algorithm 2 is designed for a small-scale texture-less scene, and its performance on the large-scale dataset is adequate.

3.3.5 Qualitative Results on Real-world Scene

Quantitative evaluation in a real-world environment is complex due to miss ground-truth data. We only conduct a qualitative evaluation procedure. The scene used to validate consists of a great many irregular metal objects, as shown in Figure 3.11. Those

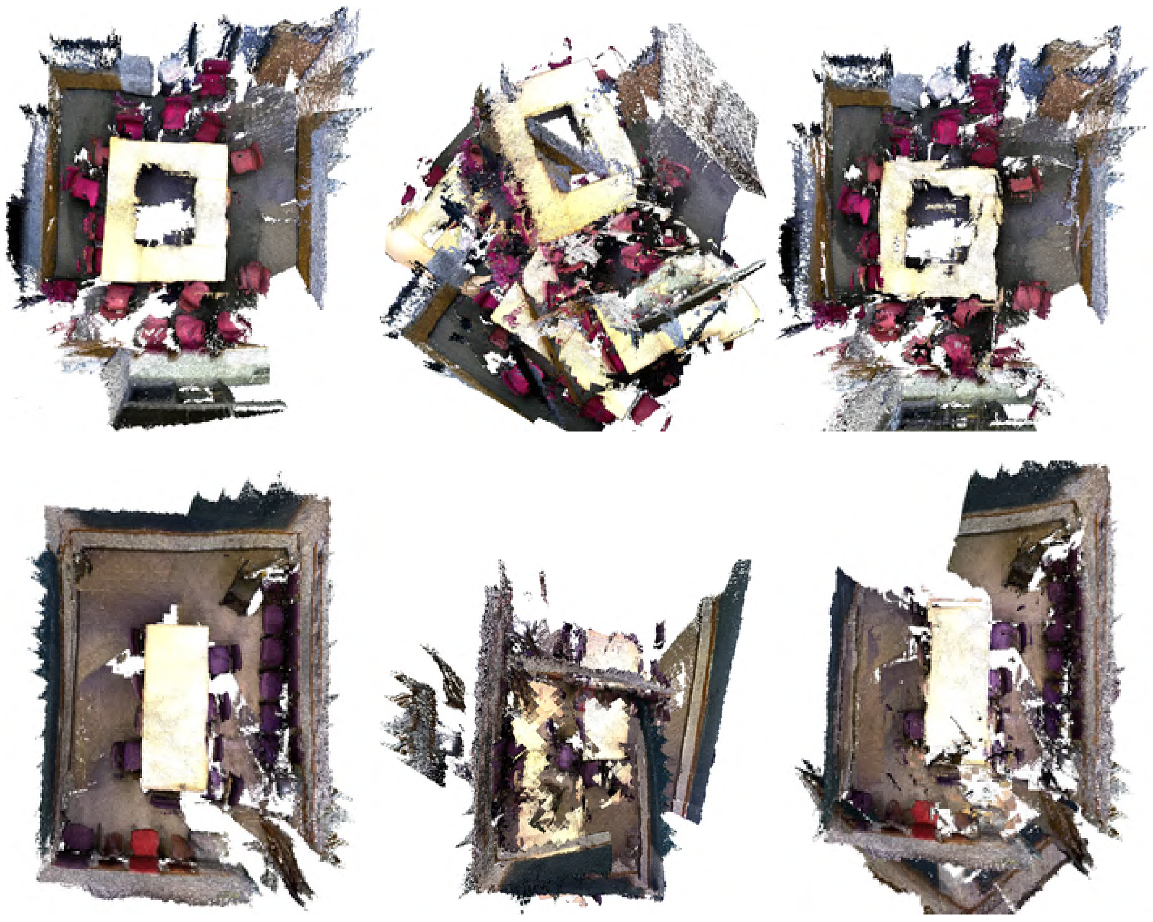


Figure 3.10: **Reconstruction of scenes from the SUN3D Dataset**. First column: ground truth. Second column: pose graph method [11]. Third column: proposed approach

metals are texture-less and small-scale with sizes ranging from 4 to 10 centimetres in length.

The input colored point cloud is captured by a stereo camera, ZED mini, with a maximum depth distance of 15 meters. The principle of the stereo camera has been introduced in section 2.1.2. In this experiment, the camera will move with a UR5 robot end-effector when scanning the environment. The results are shown in Figure 3.11. Proposed system successfully aligns the scene from the multi-view and generates a high-quality point cloud model. In this fused model, the shape and position of each

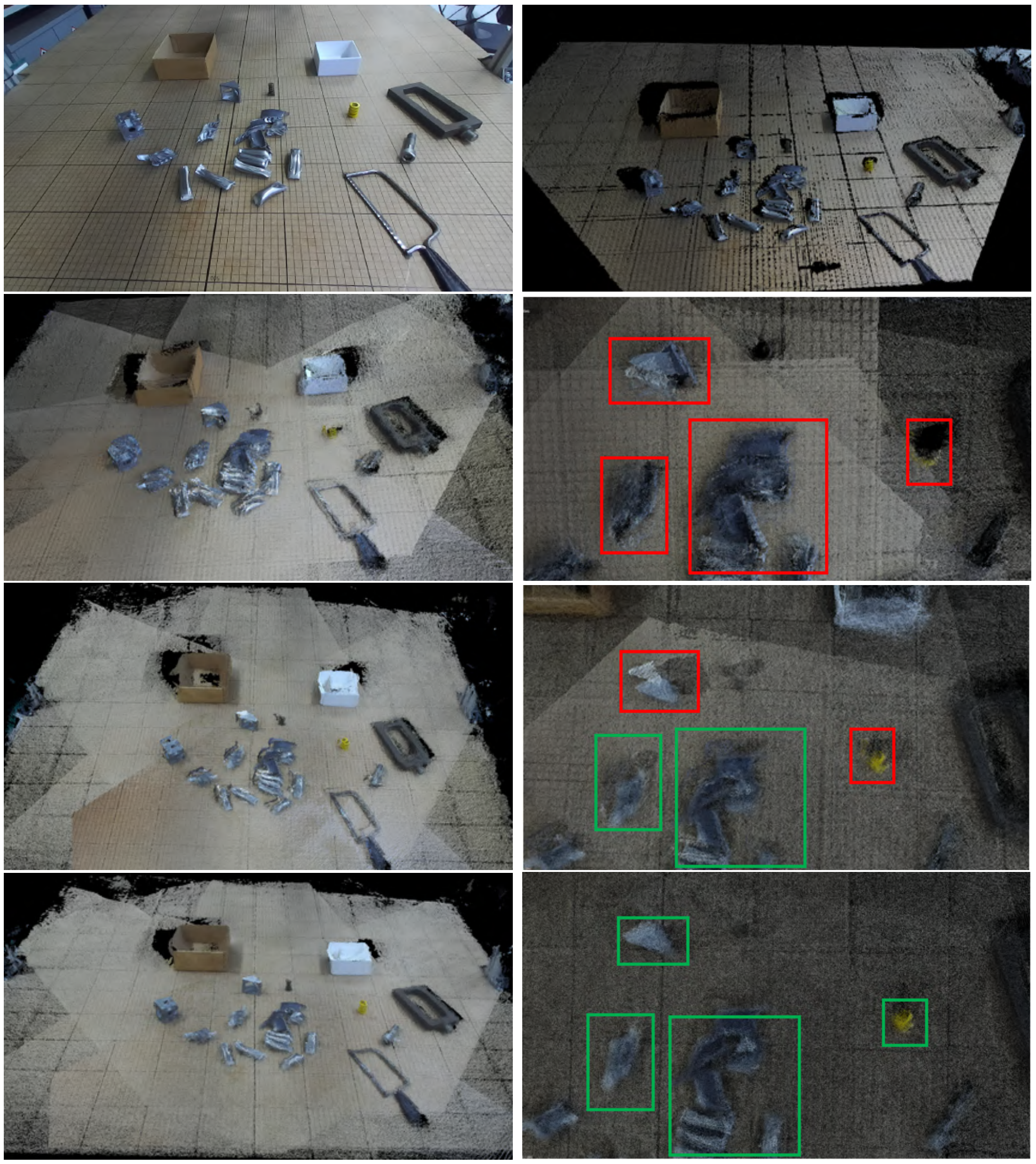


Figure 3.11: **Qualitative evaluation on real-world scene.** First row: scene image, single view point cloud. Second row: reconstructed point cloud using pose measured by ZED camera and its partial magnify. Third row: reconstructed point cloud using coloured-ICP [10] and its partial magnify. Bottom row: reconstructed point cloud using out system and its partial magnify. Red boxes indicate misaligned part and green boxes show correct alignment

object are sufficiently clear, which is a good foundation for subsequent applications.

3.4 Summary

In this chapter an improved point cloud registration method has been presented in Algorithm 1 and a reconstruction system has been illustrated in Figure 3.3. The approach fully integrates photometric and geometric information to make registration more robust and accurate as in [8]. An LM-ICP algorithm has been developed that employs the Levenberg-Marquardt method to replace the Gauss-Newton algorithm for objective optimization. To overcome the local minima and position drifting problems, a novel hierarchical coarse-to-fine strategy has been followed. Finally, we conducted evaluation experiments on both standard datasets and real-world scenes. The quantitative and qualitative evaluation of RGB-D Scenes and SUN3D datasets indicated that the improved method outperformed some widely-used ICP algorithm and a state-of-art pose graph approach. In addition, the qualitative evaluation of real-world environments shows that the improved system is effective for cluttered industrial scenarios.

Chapter 4

Image-6D: Estimating 6D Object Pose from RGB Image

4.1 Introduction

Accurate 6D pose estimation of objects is important in many real-world applications of computer vision, including augmented reality, robot manipulation and advanced autopilot operations on aerial and ground vehicles. Currently the majority of accurate 6D pose estimation methods rely on RGB-D information [100, 142, 143, 4, 15]. However, the depth sensor exposes several practical limitations such as high power consumption, limited working range, and sensitivity to the environmental effects. Such impediments mean that accurate 6D detection is not normally deployed on monocular cameras and mobile devices. The goal of this chapter is to present a precise 6D detection method that works from a single RGB image and relies on the use of deep neural networks.

Traditionally, the 6D pose estimation issue is addressed by pairing feature points between 2D images and to obtain the corresponding 3D object models [144] from the resulting cloud point. However, such approaches have failed to address texture-less targets. By contrast, the template-matching method [145, 8] is more robust than feature-matching, but it leads to low pose detection accuracy in environments full of occluded objects. Although dense feature learning approaches [146, 92] present good performance in occlusions, they fail to resolve the case of symmetric objects.

The emergence of deep learning techniques, especially CNN-based category detectors, have shown excellent outcomes for object detection [92, 20] and object segmentation [93]. Recently, there is an increasing number of works [13, 14, 97, 147], which employ deep learning for 6D pose estimation. Most of these approaches follow a similar paradigm: first they use a neural network to detect the eight 3D bounding box vertices associated with the target objects, then they perform a Perspective-n-Point (PnP) [42] algorithm calculating the orientation and translation. However, this paradigm suffers from a severe shortcoming in terms of low detection accuracy. The reason is that the key points are often not on the surface of the object, so there is some inaccuracy in the detection. As the PnP algorithm continues to accumulate these errors, an Iterative Closest Point (ICP) processing is executed in several steps to refine the pose.

The goal of this chapter is to resolve the above limitations by training a deep neural network that can accurately predict 6D pose from an RGB image in a single step. In this chapter, we propose a two-stage convolution neural network, Image-6D, inspired by Mask RCNN [93]. The next section will present the proposal in detail.

We evaluate the Image-6D (defined in section 4.2) on the LINEMOD dataset [8] (a single object 6D pose estimation dataset) and on the Occluded-LINEMOD dataset [100] (a multiple objects dataset). Additionally, we compare the result with some recent work. Furthermore, to completely evaluate the Image-6D , we perform some tests on objects in the real world.

In summary, the main contributions of this chapter are:

- We propose a novel 6D pose estimation method which can detect objects, segment instances and predict 6D pose simultaneously without any PnP process.
- We introduce Contour-Alignment, an efficient algorithm for pose refinement in an RGB image.

4.2 Definition of Image-6D

Image-6D (shown in Figure 4.1) is a RGB image-based 6D pose estimation method which consists of two main elements:

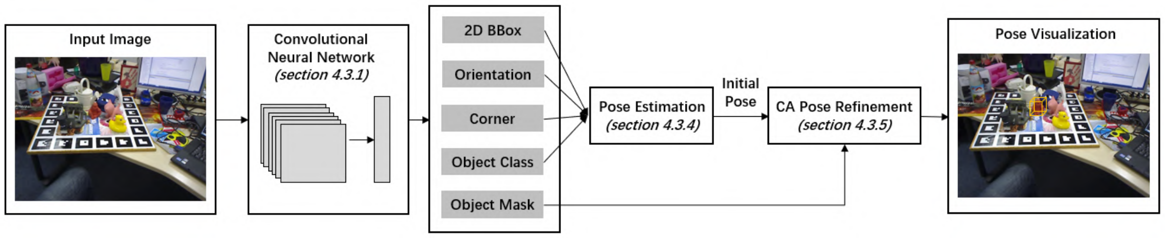

Input Image
Convolutional Neural Network (section 4.3.1)
2D BBox
Orientation
Corner
Object Class
Object Mask
Pose Estimation (section 4.3.4)
Initial Pose
CA Pose Refinement (section 4.3.5)
Pose Visualization

Figure 4.1: Image-6D: a novel learning-based object 6D pose estimation method from single RGB image.

1. Pose Estimation. Image-6D takes a single RGB image as an input and can output the object class, 2D bounding box, the object mask and object rotation simultaneously. Following these, the lateral position of the object is calculated by a reverse projection algorithm. Compared with previous works, Image-6D can estimate the object pose directly without a PnP iterative process.

2. Point Refinement. Image-6D adopts a contour alignment (CA) refinement algorithm (Algorithm 3) that align the object 2D projection and the object mask contour to replace the ICP processing. Due to the use of CA, Image-6D only needs RGB information to run.

4.3 Methodology

In this section we will introduce a novel 6D pose estimation algorithm and refinement approach. We first describe the network architecture, then we present the pose estimated method. After that, we detail the CA refinement algorithm before finally introducing the set up for training and inference.

4.3.1 Network Architecture

We propose an architecture inspired by Mask-RCNN which goes beyond Mask-RCNN in capability. This network contains two stages: i) it starts with the ResNet101 [148] backbone that extracts features over the entire image and then ii) the Region of Interest (ROI) is extracted by a Region Proposal Network (RPN) that feeds its results to the head branches. The network of Image-6D has five parallel head branches as follows:

1. class regression branch
2. box regression branch
3. segmentation head branch
4. orientation head branch
5. corner head branch

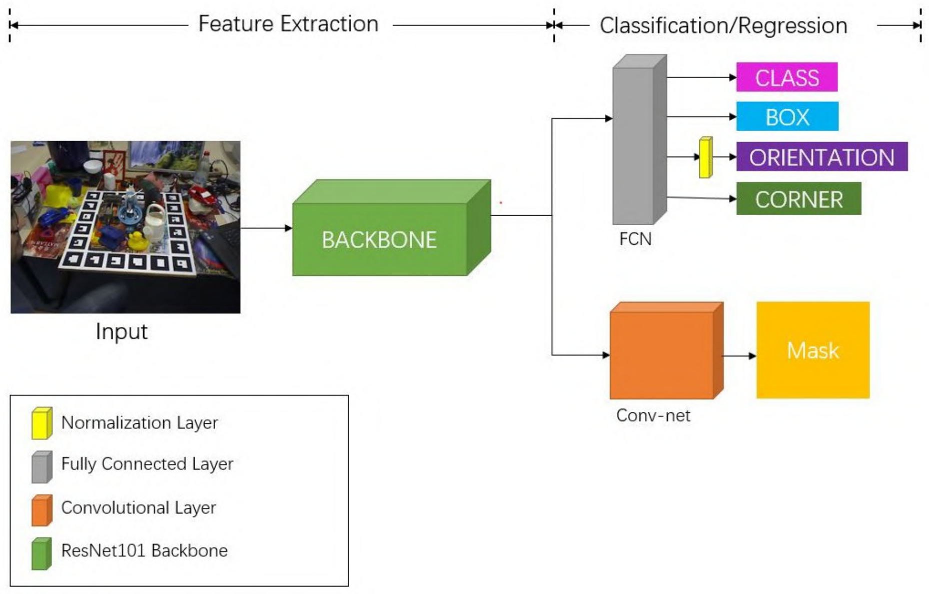


Figure 4.2: The neural network architecture of Image-6D. The shape of fully connected layers are two $4096 \times 1 \times 1$ layer and the size of convolutional layer is $7 \times 7 \times 512$.

The architecture of the neural network is shown in Figure 4.2; it takes a single RGB colour image as input and processes inputs data with ResNet101 networks. ResNet101 is a residual learning framework that is 101 layers deep. This backbone overcomes the vanishing gradient effect by utilizing a residual function. Due to its impressive outcomes, we decide to select ResNet101 as the network backbone. In addition, for the network head we firmly follow architectures proposed in Mask-RCNN to which we add a fully connected orientation prediction branch. We uses quaternions to represent orientation, so there is a normalization layer in front of a rotation layer. We also use the fully-convolutional layer to predict a pixel-wise instance segmentation by up-sampling the feature map to 28×28. Therefore, the combined network with the Mask-RCNN can achieve classification, segmentation, and estimation of 6D pose of object instances simultaneously.

4.3.2 Camera Geometric

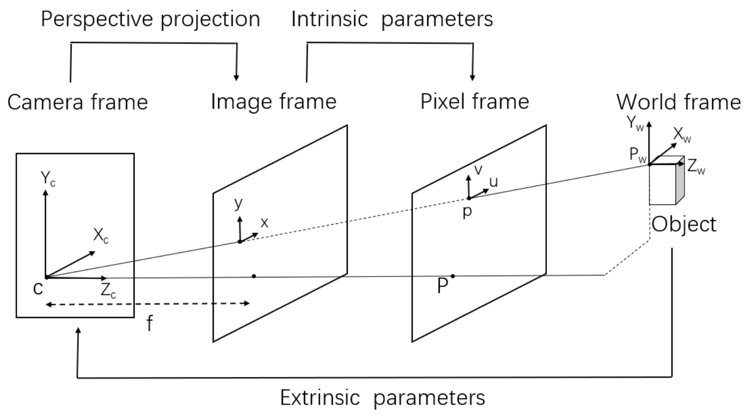


Figure 4.3: Camera projection model

Before presenting the 6D pose estimation algorithm, we first give an introduction to the basic geometry of image formation. As shown in Figure 4.3, the projection is transferring 3D point $p_w = [x, y, z]^T$ in camera coordinates to 2D point $p = [u, v]^T$ in pixel coordinates. Firstly, transforming 3D points p_w in the world coordinate system into the camera coordinate p_c by the camera extrinsic parameter C that composed of a 3×3 rotation matrix R and a 3×1 translation matrix t:

$$p_c = \boldsymbol{R} p_w + \boldsymbol{t} \tag{4.1}$$

And then we can represent the projection of the point P_C at camera coordinate on $(u, v)^T$, this processing is also referred to as perspective projection.

$$x = \boldsymbol{f} \frac{x_C}{zc} \qquad y = \boldsymbol{f} \frac{y_C}{z_c} \tag{4.2}$$

Using homogeneous coordinates to write eqn. (4.1) and eqn. (4.2) as:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix} \qquad x = \frac{x'}{z'} \qquad y = \frac{y'}{z'} \tag{4.3}$$

Subsequently, we use intrinsic parameters to transform the projected image on image coordinate to the pixel coordinate, this conversion as known as affine transformation.

$$u = x + C_x \qquad v = y + C_y \tag{4.4}$$

Taking into account all of these effects, we can obtain a camera intrinsic matrix K:

$$K = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

where f_x and f_y are the focal distance along x axis and y axis, and C_x and C_y are the principal points in the centre of the image.

Overall, the camera formation describes the projection of 3D world points into 2D pixel coordinates, in such a way that the entire process can appear concisely as:

$$p = K[R|t]p_w \tag{4.6}$$

4.3.3 Perspective-n-Point

The Perspective-n-Point (PnP) problem is proposed by Fischler and Bolles [41] to obtain the pose of a calibrated camera according to a group of n 3D points in world coordinate and their 2D projection points in the image coordinate. PnP techniques have numerous applications in computer vision [149] and photogrammetry [46]. As we reviewed in Chapter 2, correspondence-based 6D pose estimation [94, 14, 23, 96] commonly use a PnP algorithm or its variant to compute the 6D pose of an object. In this section, we introduce a basic PnP algorithm even though we haven't adopted PnP in Image-6D. It is still an important concept in the field of 3D vision. Besides, it is convenient to explain the reasons for not adopting PnP by introducing it. More importantly, a CA refinement algorithm involves a similar iteration solution.

In the last section, we present how to project a 3D point in world coordinates into image coordinates by utilization of a camera matrix. While the PnP problem is given 3D points, 2D projection points and camera intrinsic matrix to calculate camera pose, also known as extrinsic camera parameters, $[\mathbf{R}|\mathbf{t}]$. To solve the PnP problem, we need to knowthe location of at least 3 points. Specifically for 6D pose estimation, it usually needs 8 points representing 8 corner points of 3D objects. The following part presents a widely used PnP solver, EPnP [42]. The main step of EPnP is:

1. Select n = 4 control points in world coordinates and convert them to image coordinates, obtaining the 3D control point p_i^w, their corresponding projection 2D points p_i^c. The detail of control point selection can be found in [42].

2. Calculate the centroid of p_i^w, c^w, and matrix A.

$$c^w = \frac{1}{n}\sum_{i=1}^{n} p_i^w \tag{4.7}$$

$$A = \begin{bmatrix} p_1^{wT} - c^{wT} \\ ... \\ p_n^{wT} - c^{wT} \end{bmatrix} \tag{4.8}$$

3. Calculate the centroid of p_i^c, c^c, and matrix B.

$$c^c = \frac{1}{n}\sum_{i=1}^{n} p_i^c \tag{4.9}$$

$$B = \begin{bmatrix} p_1^{cT} - c^{cT} \\ ... \\ p_n^{cT} - c^{cT} \end{bmatrix} \tag{4.10}$$

4. Calculate the matrix H.

$$H = B^T A \tag{4.11}$$

5. Perform the SVD decomposition [150] of matrix H.

$$H = U \sum V^T \tag{4.12}$$

6. Estimate rotation matrix **R**.

$$R = UV^T \tag{4.13}$$

7. Compute translation vector **t**.

$$\boldsymbol{t} = c^c - \boldsymbol{R}c^w \tag{4.14}$$

8. Mnimize the reprojection error ***err*** via Gauss-Newton algorithm to refine the **R** and **t** until meet termination criteria or reach maximum iteration number.

$$\boldsymbol{err} = \|p^c - K(\boldsymbol{R}p^w + \boldsymbol{t})\| \tag{4.15}$$

The parameters A, B, H, U and V are intermediate variables derived from the calculation process, so we don't notate for them. In actual usage, we don't need such complicated steps. The PnP solver can be used directly from some open-source libraries such as OpenCV. However, Chen et al. [151] notices that such PnP-based methods will fail when the target object is truncated. Do et al. [152] argue that, this strategy that predicts corner vertices and then leverages PnP to estimate 6D pose, is time-consuming for inference, especially when the predicted vertices and initial model points are far away. In the next section, we will present the novel pose estimation method without PnP processing.

4.3.4 Pose Estimation

In last section we introduce a forward camera projection. While in this section, we will use a backward projection to recover 3D information from the 2D pixels.

The object pose usually includes a rotation matrix and a translation vector. The rotation matrix is estimated using quaternion regression from the neural network. As for the translation vector, instead of predicting it from neural networks directly, we have designed a fast and simple algorithm to calculate it. The reason why we deprecate regression of translation is that the neural network can't handle camera intrinsic matrix changes. It is impossible to train a network for each type of camera. So the Image-6D predicts object rotation and translations separately.

As shown in Figure 4.4, the translation vector $t = [t_x, t_y, t_z]$ defines the coordinates of the object center in the camera coordinate system and the cat model is under the current orientation. The crucial step to estimate the 3D translation is calculating t_z. The camera projection is a 3D-to-2D perspective projection, and we utilize a reverse projection principle to recover the depth t_z in this system. In section 4.3.1, we introduced the neural network architecture that can output an initial orientation with quaternion representation. Additionally, object 3D model is known. As a result, a transformed model can be obtained through multiply rotation matrix and object model. Eqn.2.2

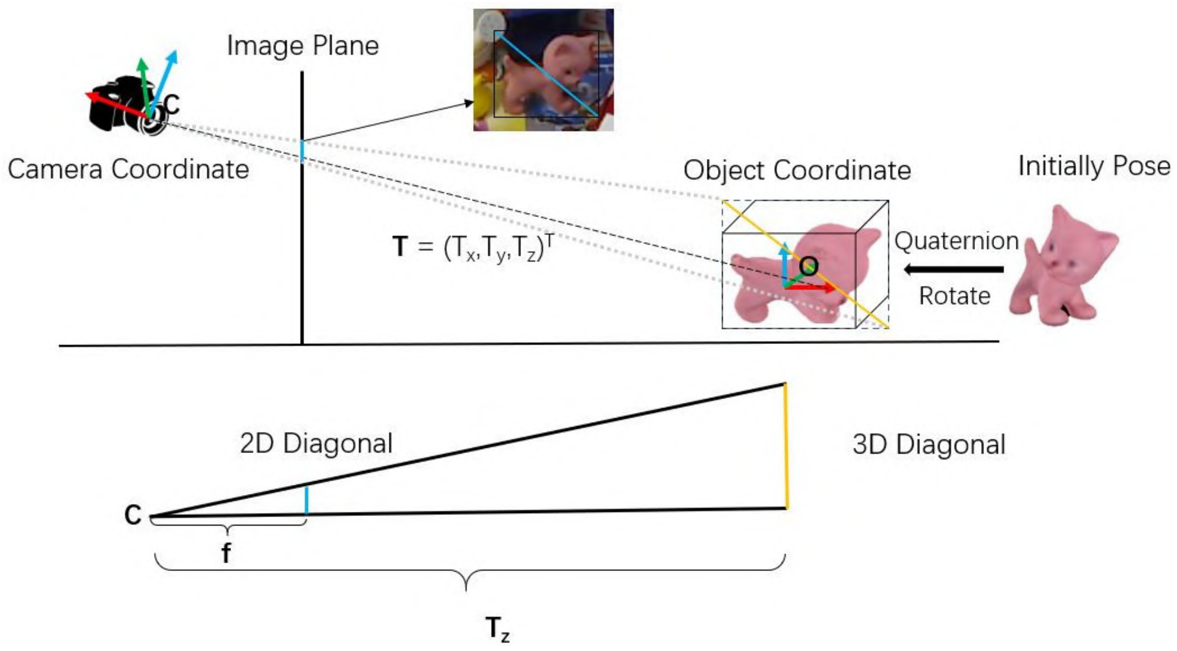


Figure 4.4: Illustrating the relationship between the object coordinate system and the camera coordinate system. The 3D translation is calculated by a projection principle.

represents the conversion between quaternion and rotation matrix.

As illustrated in Figure 4.4, a 2D diagonal (the blue line obtained from neural network) and a 3D diagonal (the yellow line calculated by 3D model and quaternion) can be used to derive the t_z:

$$t_z = \frac{3Ddiagonal}{2Ddiagonal} * f \tag{4.16}$$

where f denote the focal lengths of the camera. We assume that the focal lengths in horizontal f_x and vertical f_y directions are equivalent. The same procedure can be easily adapted to obtain t_x and t_y:

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \frac{u - c_x}{f_x} * t_z \\ \frac{v - c_y}{f_y} * t_z \end{bmatrix} \tag{4.17}$$

where $[u, v]$ is the object center that predicts from the neural network, $[c_x, c_y]$ expresses the principal point, which would be theoretically in the centre of the image.

4.3.5 Pose Refinement

Though the estimated object poses are already precise, they can still be improved by a further refinement. For the RBG-D data, the detection is usually followed by ICP processing. In this paper, we propose an edge-based refinement algorithm by aligning the object instance contours and 2D projection contours. We call it a contours alignment (CA) algorithm. This method can be adapted to any CNN-based 6D pose estimation framework to improve accuracy. Note that this refinement processing is to eliminate the error in translation. For the rotation error, we don't consider the CA refinement procedure optimal because the rotation predicted by the network is accurate enough and it will increase refinement time if we consider both translation and rotation.

Algorithm 3 Position Refinement

```
Input: Initialise object pose $P_0$; Object mask $M_0$ predicted by neural network; 3D-2D
    projection function $f$; Object model;
Output: Refine object pose $P_n$
1: Calculate contour $C_0$ for object mask $M_0$.
2: Set $C_0$ as reference points.
3: Compute 2D projection $proj$ with current pose $P_0$, $proj = f(P_0)$
4: Extract contour $C_1$ from $proj$.
5: Apply a closest point pairs algorithm between $C_0$ and $C_1$ to obtain $C_3$.
6: Compute residual error: $f_r = C_3 - C_1$.
7: Calculate Jacobian matrix $J$ of $f$, so $df = Jdx$.
8: Solve $\Delta t$ using pseudo inverse $\delta t = (J^T J)^{-1} J^T f_r$, and update pose $P_0$.
9: Repeat steps 3-8 until reach threshold; return $P_0$.
```

In Algorithm 3, we extract contours by using the "find_contours" function from the skimage module [153], that is an image processing module in python. The "find_contours" function uses the “matching squares” and linearly interpolated approach to obtaining the iso-valued contours of the input 2D array for a specific value. The closest points pairing process employs a kd-tree search from the "sklearn.neighbors" module [154]. The closest point pairs guarantee that two contour arrays have the same shape so that we can perform arrays subtraction.

In order to achieve an appropriate balance between accuracy and efficiency, we only optimize the translation because the error in translation is more dominant than rotation. The Jacobian matrix J is:

$$J = [\frac{\partial f}{\partial t_x}, \frac{\partial f}{\partial t_y}, \frac{\partial f}{\partial t_y}] \tag{4.18}$$

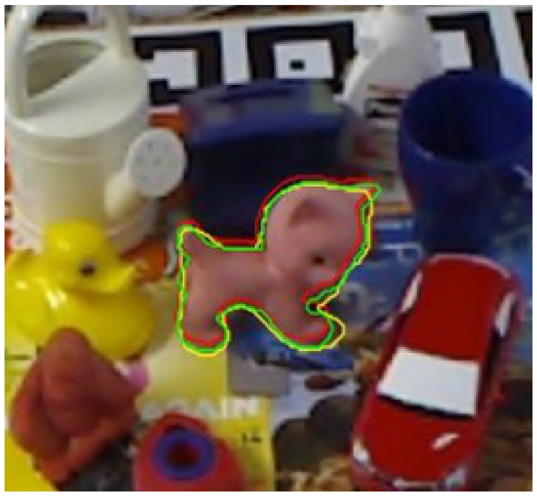
we approximate the derivatives to obtain:

$$J \approx \begin{bmatrix} \dfrac{f(t+[\epsilon,0,0]) - f(T)}{\epsilon} \\ \dfrac{f(t+[0,\epsilon,0]) - f(T)}{\epsilon} \\ \dfrac{f(t+[0,0,\epsilon]) - f(T)}{\epsilon} \end{bmatrix}^t \tag{4.19}$$

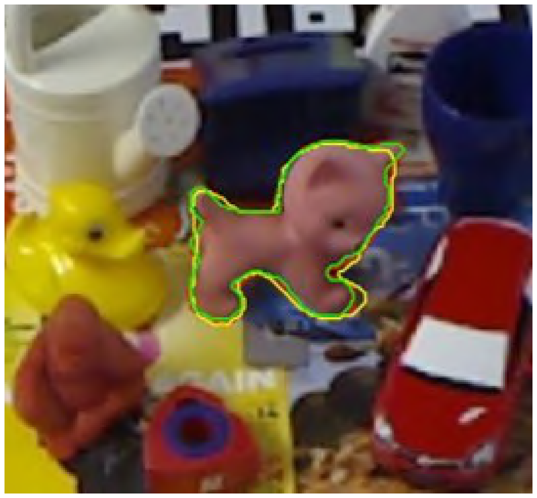
where t denotes the translation vector and ϵ is a tiny number. In this paper we choose $\epsilon = 0.0000001$ to guarantee the size of projection points is constant. Therefore, in Algorithm 3, the two contour arrays C_1 and C_3 can subtract because they have same size. For the solution of increment Δt, we use the Gauss-Newton method introduced in section A.1. This alignment for contour is not complicated so we choose the Gauss-Newton method rather than Levenberg-Marquardt method. This is because Gauss-Newton is easy to code and runs faster.

In Figure 4.5 one can observe that the projection contour extracted by refinement of pose (red line) coincides with the ground truth contour (green line). This shows that CA refinement Algorithm 3 can improve pose accuracy significantly. Furthermore, we can also see both the translation error and the pixel error being reduced by nearly 60% in the first refinement iteration and moreover this tends to converge after the second refinement iteration. Therefore, Algorithm 3 can refine object pose quickly and effectively.

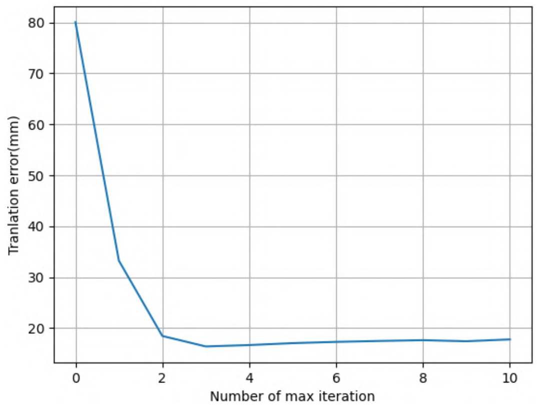
Essentially, CA refinement Algorithm 3 and the PnP algorithm mentioned in section 4.3.3 all belong to the 2D-3D correspondences problem. But the difference is that: 1)



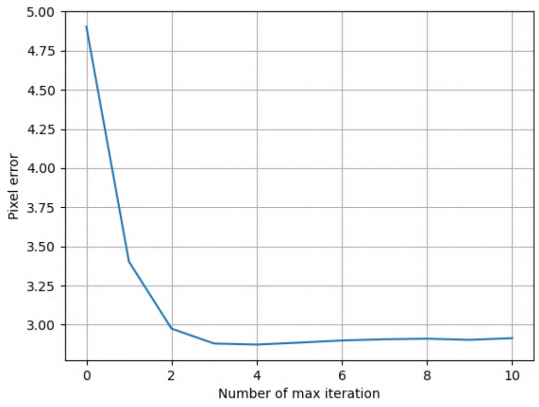
(a) 2D Projection contour without refinement.



(b) 2D Projection contour after refinement.



(c) Translation error under different iteration times.



(d) Pixel error under different iteration times.

Figure 4.5: Improvement of CA refinement algorithm 3 for 6D pose estimation. In (a) and (b), green lines show the ground truth contour, yellow lines present the predicted mask contour and red lines indicate 2D projection using the current pose.

Algorithm 3 has obtained a rough 6D pose as the initial pose while the initial pose of PnP is random; 2) PnP requires control points as references, and CA refinement relies on closet points and 3) The optimization strategy is different.

4.3.6 Training and inference

We have implemented the proposed network Image-6D in Python3 using the TensorFlow library [155]. The input to the neural network was an RGB image with size 640×480. The training data consisted of three parts: first is the RGB image; second is a binary mask image and the third part is a label. Unlike other approaches using eight corner annotations or 6D pose annotations, we adopt a new annotation method based on a quaternion and two corner points as shown in the Figure 4.6, because such an annotation can fit the proposed pose estimation algorithm (shown in Figure 4.4) better.

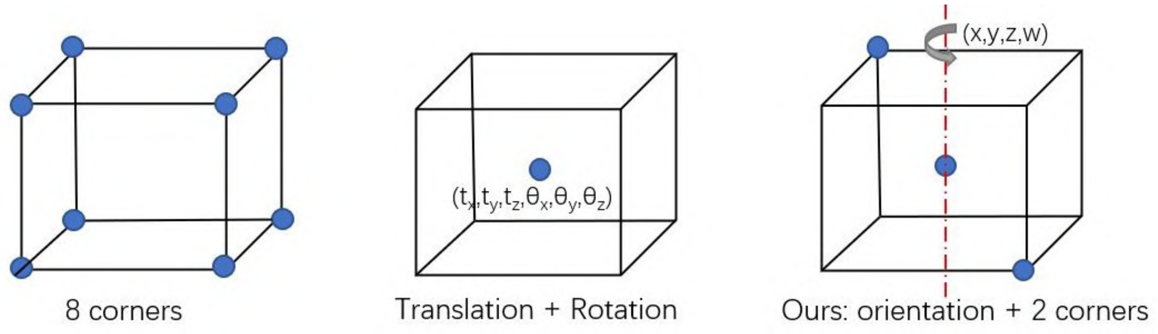


Figure 4.6: Comparing with different annotation method

In training, we define a multi-task loss to jointly train the classification, bounding box regression, instance segmentation, quaternion regression and corner point regression. Formally, the total loss function is defined as follows:

$$L = \alpha_1 L_{cls} + \alpha_2 L_{box} + \alpha_3 L_{mask} + \alpha_4 L_{quat} + \alpha_5 L_{cor} \tag{4.20}$$

where α_1, α_2, α_3, α_4, α_5 are loss weights, which indicate the importance of each loss component. In experiments, we set $\alpha_1 = \alpha_2 = \alpha_5 = 1$, $\alpha_3 = 10$ and $\alpha_4 = 2$. L_{cls} is softmax loss, L_{box} and L_{cor} are smooth L1 loss, L_{mask} is binary cross-entropy loss, and L_{quat} is a derivation of L2 Loss, defined as follows:

$$L_{quat} = \frac{\sum_{i=1}^{n} (\beta r_i - \beta \bar{r})^2}{n} \tag{4.21}$$

where r_i denotes the predicted quaternion and $\bar{r}$ is the ground true quaternion. The four parameters of the quaternion are all between 0 and 1, so we apply a magnification factor β ($\beta = 10$ in experiments).

We train the neural network on a Tesla V100 GPU for 90 epochs. The first 20 epochs train network heads with a 0.002 learning rate. Then, using the same learning rate, we fine tune the layers from ResNet stage 4 in the next 10 epochs. After that, we train all layers for 30 epochs. In the following 10 epochs, the learning rate is decreased by 10 until we train all the layers. Lastly, we change the learning rate to 0.00002 to fine tune all the layers in the final 10 epochs.

At the inference phase, we select object instances which have their detection scores higher than 0.9. Proposed pose estimation method presented in section 4.3.4 and refinement Algorithm 3 are then applied to the detected objects to obtain accurate 6D pose matrices.

4.4 Experiments

We conduct experiments on two standard data sets including a single object pose data set LINEMOD, and a multiple objects pose data set Occlusion-LINEMOD to evaluate the Image-6D (introduced in section 4.2) for 6D pose estimation. We compare Image-6D against some widely used state-of-the-art 6D pose estimation approaches. We also prove that Image-6D can apply to real-world custom objects.

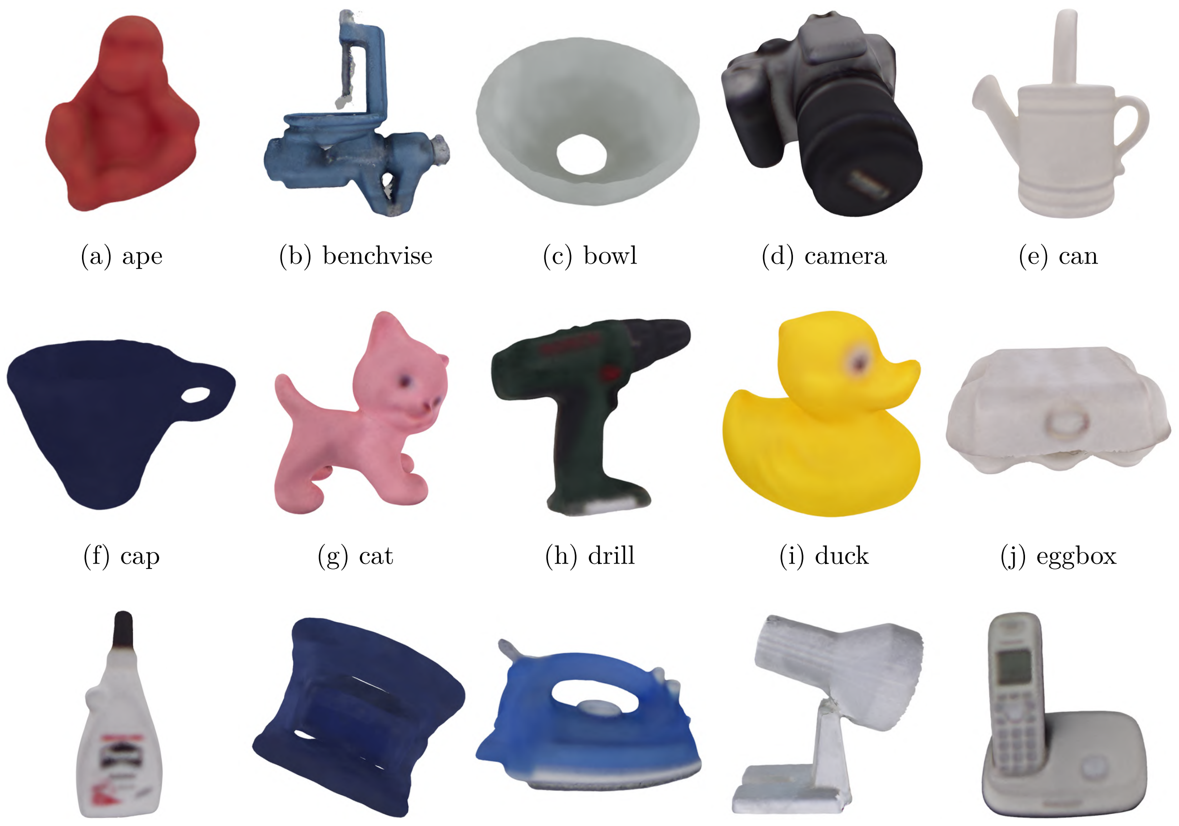

(a) ape (b) benchvise (c) bowl (d) camera (e) can
(f) cap (g) cat (h) drill (i) duck (j) eggbox
(k) glue (l) holepuncher (m) iron (n) lamp (o) phone

Figure 4.7: Objects list in LineMOD dataset.

4.4.1 Metrics

Image-6D is evaluated under the average distance (ADD) metric [8]. The average distance calculates the mean of pairwise distances between 2D projections of the 3D models, calculated utilizing the estimated pose and ground truth pose:

$$ADD = \frac{1}{m}\sum_{x\in M}\min_{M}\|(\boldsymbol{Rx}+\boldsymbol{T})-(\bar{\boldsymbol{R}}\boldsymbol{x}+\bar{\boldsymbol{T}})\| \tag{4.22}$$

where $\boldsymbol{R}$, $\boldsymbol{T}$, $\bar{\boldsymbol{R}}$, and $\bar{\boldsymbol{T}}$ are ground true rotation, ground true translation, estimated rotation and estimated translation, respectively. M denotes the vertex set of the 3D model, and m means the number of 3D points. Evaluation is based on the widely used metric **ADD-0.1d** and **REP-5px**, where the estimated pose is considered to be correct if the average distance is below 10% of the object's diameter or smaller than a

5 pixels threshold.

4.4.2 Evaluation On LINEMOD Dataset

We first test Image-6D (that is section 4.2) on the LINEMOD dataset, which contains 15 objects with poor texture shown in Figure 4.7 in a cluttered environment. In common with other papers in the literature, we evaluate methods on 13 of these objects. We adopt similar settings with [14] to randomly select 30% of the images as training data and the rest of images as test data. However, only RGB images are used in the training and testing phase.

Table 4.1: Comparison of Image-6D with state-of-the-art work on LINEMOD data set in terms of ADD-0.1 metric. We present percentages of correctly estimated pose and highlight the best result among those by **bold** numbers.

Object / **Method**	Zhao	Yolo-6D	SSD-6D	Image-6D
Ape	35.1	21.62	0	**42.29**
Benchvise	23.9	**81.8**	0.18	77.64
Cam	33.2	36.57	0.41	**66.78**
Can	21.0	68.80	1.35	**74.09**
Cat	30.6	41.82	0.51	**57.89**
Driller	28.6	63.51	2.58	**70.45**
Duck	27.9	27.23	0	**37.81**
Eggbox	38.9	**69.58**	8.9	64.5
Glue	31.2	**80.02**	0	44.51
Holepuncher	13.4	42.63	0.30	**62.40**
Iron	37.8	74.97	8.86	**78.01**
Lamp	34.5	71.11	8.20	**84.5**
Phone	19.9	47.74	0.18	**65.27**
Average	28.9	55.95	2.42	**63.59**

We compare Image-6D with the state-of-the-art approaches Yolo-6D [14], Zhan [156] and SSD-6D [13], which run under a similar setting. In Table 4.1, the results for the competing methods are presented. On average, Image-6D outperforms all the considered competitors by a margin of at least 7% or more. We also find that Image-6D is more effective for small-size objects. For example, with the camera model whose diameter is 17.24 cm, the estimated pose accuracy increases by nearly 30%. Even when compared with some RBG-D based methods such as SSD-6d, for which the average accuracy reaches 76.3%, Image-6D is still competitive.

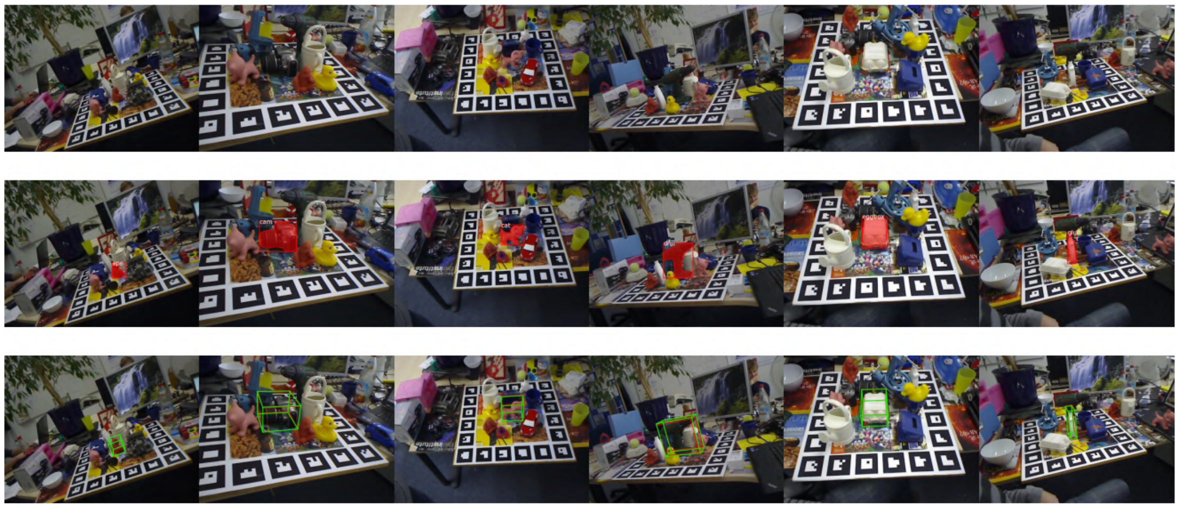


Figure 4.8: **Qualitative results on LINEMOD.** First row : the original images. Second row: the predicted object class, 2D bounding box and segmentation. Third row: 6D pose represented by 3D bounding boxes which green is the ground truth and the red is estimated.

Table 4.2 lists overall inference time of Image-6D. This model runs at 8.115 frames per second (fps) on an Nvidia Tesla V100 GPU without refinement, which equals 123 ms per image. With refinement, our approach can run at 1.82 fps, which is 549 ms per image. It is worth noting that the object model's size impacts refinement time. For the small size objects (e.g. ape and duck), the inference time is over 2 fps. However, for some large objects, such as driller, the inference time only have 0.81 fps.

Table 4.2: Overall computational runtime of our approach (FPS)

Object / **Method**	Ape	Can	Cat	Driller	Duck	Eggbox	Glue	Holepuncher	Average
w/o Refinement	8.16	7.48	7.97	8.19	7.78	8.63	8.19	8.52	8.115
w/ Refinement	2.27	1.14	1.94	0.81	2.17	2.62	1.79	1.83	1.82

4.4.3 Evaluation On Occlusion-LINEMOD

The Occlusion-LINEMOD is a multi-objective estimation benchmark which contains 8 objects and 1214 images (see Figure 4.7). As its name shows, a few objects in the images are heavily occluded, which makes estimation extremely difficult.

To create training data, we follow the same data selection setting as in the previous evaluation. Due to every image containing several instances, the network needs more training epochs to reach convergence, so we modify the training strategy: the training epoch increases from 90 to 160. The reason we select 160 epochs is that the network can reach convergence without consuming a lot of time. The first 20 epochs train network heads with 0.004 learning rate. Then, using the same learning rate, we fine tune layers from ResNet stage 4 and up during the next 10 epochs. After that, we train all layers for 70 epochs. This initial learning rate value can make training convergence quick. In the next 20 epochs, the learning rate is decreased by 10 in all layers. Finally, the learning rate is set to 0.00004 in order to fine tune all layers in the final 20 epochs. Through twice learning rate tuning, we can minimize loss. This setting achieves excellent performance in experiments. In addition, the segmentation loss weight α_3 changes to 40 in order to overcome excessive occlusion in the image.

As can be seen from the Table 4.3, Image-6D outperforms other methods, such as PoseCNN [4], Heatmaps [127], Seg-drive [97], iPose [128], Yolo-6D [14], in both the ADD-0.1d metric and REP-5px metric. In Figure 4.9, we can notice that the estimated pose is still accurate with partial occlusion. But if the visibility of the object is too low, the estimation will fail.

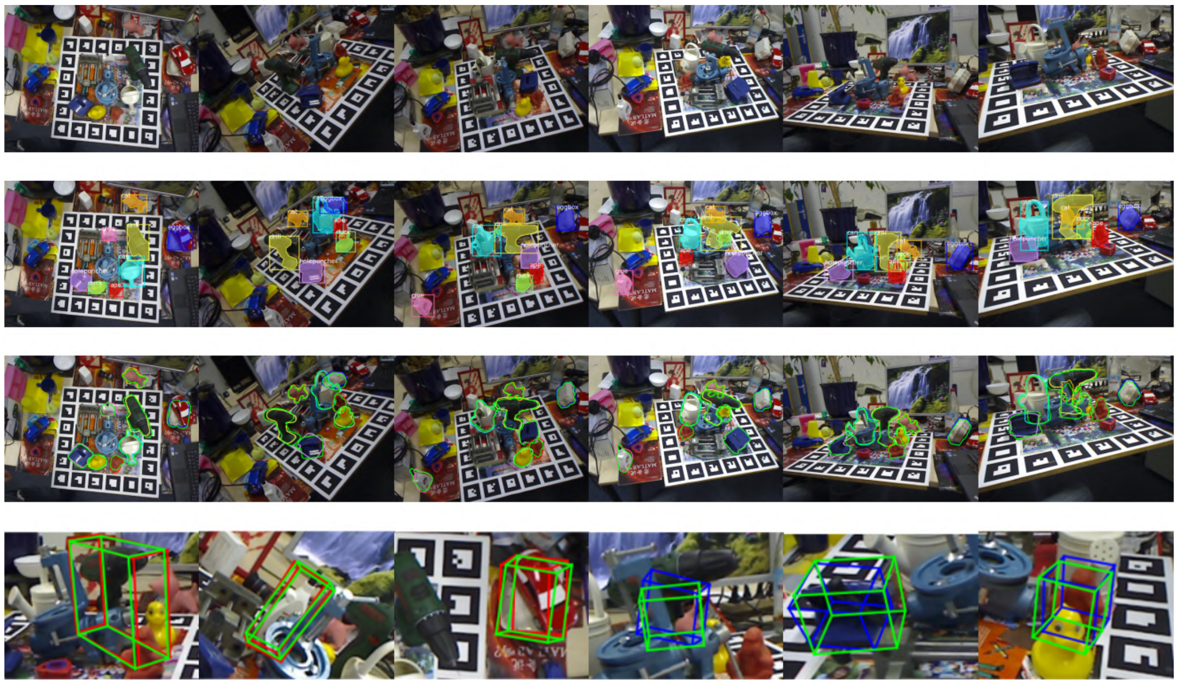


Figure 4.9: **Qualitative results on Occluded LINEMOD.** First row : the original images. Second row: the predicted object class, 2D bounding box and instance segmentation(different color means different class). Third row: 6D pose represented by 2D projection contour which green is the ground truth and the other color is estimated. Fourth row: Area screenshot, the first three columns is success cases and the last three columns is fail cases.

Table 4.3: Comparison of Image-6D with state-of-the-art works on Occluded LINEMOD dataset in terms of ADD-0.1 metric and REP-5px metric. We present percentages of correctly estimated pose and highlight the best result among those by **bold** numbers.'-' denote the results not in the original paper.

Method / **Object**	ADD-0.1				REP-5px		
	PoseCNN	Heatmaps	Seg-drive	Image-6D	iPose	Yolo-6D	Image-6D
Ape	9.6	16.5	12.1	**18.87**	24.2	7.0	**54.69**
Can	45.2	42.5	39.9	**50.52**	30.2	11.2	**44.82**
Cat	0.9	2.8	8.2	**15.38**	12.3	3.6	**53.73**
Driller	41.4	**47.1**	45.2	34.0	-	1.4	**17.49**
Duck	19.6	11.0	17.2	**27.00**	12.1	5.1	**51.91**
Eggbox	22.0	**24.7**	22	20.62	-	-	**41.37**
Glue	38.5	**39.5**	38.5	26.43	25.9	6.5	**43.72**
Holepuncher	22.1	21.9	**36.0**	32.0	20.6	8.3	**31.78**
Average	24.9	25.8	27.0	**28.1**	20.8	6.2	**42.43**

4.4.4 Qualitative Results on Real-world Image

The object models in the standard data set are precise, and the annotations are accurate. However, in the real world, it is hard to obtain a perfect object model and annotate poses on authentic images. Obtaining ground truth data is time-consuming, error-prone, and costly [157]. Besides, current methods for annotating real world data [8, 4] are not scale so they can't generate the massive data for network training. Therefore, we consider synthetic images to train so that this method can apply Image-6D on a broader range of objects.

A synthetic image data is generated by rendering object models with a random background. Theoretically, we can obtain infinite image data with arbitrary poses. Besides, to bridge the visual reality gap, complex backgrounds, noise and obstacles are considered to augment data.

In experiments, the object model shown in the Figure 4.10 (b) is obtained by a structure from motion (SFM) [46] method, which can reconstruct an object model

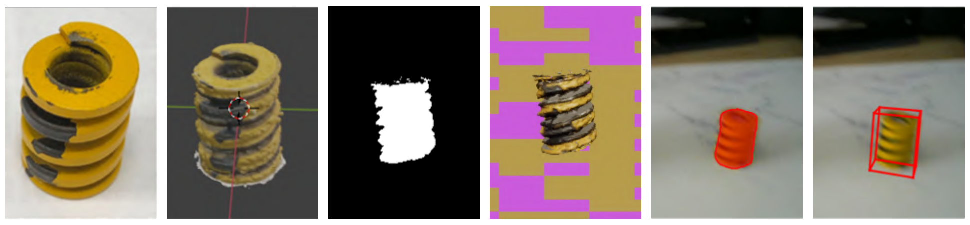


Figure 4.10: The application in real world object: (a) Real object. (b) Object model. (c) Synthesis mask. (d) Synthesis RGB image. (e) Detected mask. (f) Estimated pose.

using the object images illustrated in Figure 4.10 (a) captured from multi-view. Then, utilizing the NVIDIA Deep learning Dataset Synthesizer (NDDS) tool [29] generates synthetic training data. Figure 4.10 (c) shows the synthetic mask of the object, and Figure 4.10 (d) is the RGB image generated by the NDDS tool.

Afterwards, the training data is organized as section 4.3.6 presented: 1) synthetic RGB image; 2) synthetic binary mask image and 3) label. All of those three parts can be acquired from the NDDS tool output results. Finally, we feed the organized training data into the proposed neural network as Figure 4.2 shown. Figure 4.10 (e) and (f) are the visualization of the output in the real test image, of which (e) is the mask output and (f) is the 3D bounding box output. We can note that the performance is not getting worse due to the use of synthetic data. Therefore, the pipeline of 6D object pose estimation can be more generic using synthetic data.

4.5 Summary

In this chapter, we have introduced a new method Image-6D to detect an object class, segment instance and estimate object 6D pose simultaneously from a single RGB image. This method can predict object orientation and calculate translation without a PnP process. What's more, we propose a novel pose refinement algorithm Contour-Align shown in Algorithm 3 aligning the mask contour and the 2D projection contour for the single RGB image. This refinement technique can be applied to most of the post-processing of RBG based 6D estimation. Furthermore, the evaluation shows the accuracy of the Image-6D surpasses current state-of-the-art methods [13, 14, 127, 128, 97] in LineMOD and Occlusion-LineMOD dataset under the ADD-0.1d metric. Therefore, this work is encouraging because it indicates that it is feasible to accurately predict the 6D pose object pose in a cluttered environment using RGB data only. An interesting future work is to improve the estimation accuracy when the CAD model is unavailable.

Chapter 5

Point-6D: Estimating 6D Object Pose from Point Cloud

5.1 Introduction

In the last chapter, we have introduced a 6D pose estimation method using a single RGB image. The development of low-cost RGB-D sensors has enabled 6D object pose estimation systems which are more reliable and robust than RGB-only methods in a poorly-lit scene. Typical RGB-D based 6D pose estimation employs CNN to extract learning features from the RGB channel and depth channel, respectively, and then fuses two features feeding to the next stage [158, 159, 103]. However, the state-of-the-art techniques struggle in how to better utilise the complimentary nature of color and depth information.

In the field of autonomous driving, Frustrum PointNet [103] and PointFusion [104] achieve the high quality 3D detection for pedestrians, cyclists, and cars via full advantage of point cloud and image information in a heterogeneous network. Inspired by those works, Wang et al. [15] proposed DenseFusion, which fuse and embed RGB pixel and point cloud at a per-pixel level. Besides, through an additional trainable iterative refinement procedure, DenseFusion significantly improves the performance of a RGB-D based 6D pose estimation method in Linemod [3] and YCB-Video [4] dataset. On the basis of per-pixel prediction, He et al. [9] presented PVN3D, a dense correspondence method utilizing 3D hough voting.

In an existing system, some authors [15, 4, 103] only consider to fuse point cloud and color image information. In fact, point clouds also have significant properties and normals, which can represent the relationships between a point and it's neighbors. In this work, we propose Point-6D, a novel deep neural network that estimates the 6D pose of a known object using a colored point cloud as the input. The next section will introduce the proposal in detail.

We further conduct evaluation on two widely-used 6D pose estimation datasets, Linemod [3] and YCB-Video [4]. Evaluation results indicate that Point-6D outperforms some state-of-the-art systems in detection accuracy.

In summary, the main contributions are listed as follow:

- We propose a novel deep neural network Point-6D algorithm, which can merge point, color and normal features from three channels to regress the 6D pose of known objects.
- We develop a jointed loss function to make the network more effective and easy to train.
- We demonstrate the performance of Point-6D and state-of-art 6D pose estimation system on the Linemod [3] and YCB-Video [4] datasets.

The following part of this chapter moves on to describe in greater detail the network architecture and evaluate an experiment. section 5.2 defines the Point-6D method, section 5.3 presents the proposed system, section 5.4 introduces the experiments in detail and section 5.5 summarizes the outputs of the work.

5.2 Definition of Point-6D

Point-6D is a point cloud-based 6D pose estimation system which consists of three components shown in Figure 5.1:

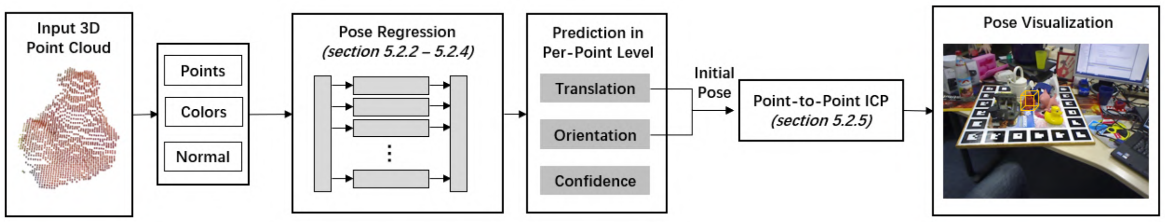


Figure 5.1: Point-6D: a novel learning-based object 6D pose estimation method from colored point cloud

1. Data Preparation. Point-6D extracts the points location, colors and normal in per-point level from input 3D point cloud.
2. Pose Regression. Point-6D employs multi-layer perceptron (MLP) to extra features from points, colors and normals respectively, and then merge them as a global features as per-point level. Afterwards, we develop a pose net to regress translation offset, orientation represented by quaternion and confidence for each point. The initial 6D pose is selected by the highest confidence scoring. The benefit of prediction at point level is that it allows detection to perform well even in a heavily obscured environment.
3. Point Refinement. Point-6D uses a point-to-point ICP (that is Algorithm 4) to refine the initial pose.

5.3 Methodology

In the last chapter, we introduced a novel 6D pose estimation method using a RGB image only. It mainly applied in the scenarios that depth information is not available or the computational power is low e.g. wearable devices, smartphones, etc. But the accuracy of RGB image-based 6D pose estimation cannot apply in tasks requiring precise positioning, such as robotic manipulation. The goal of the work in this chapter is to fully utilize the geometric and photometric information of the point cloud predicting 6D pose of a set of known objects in a cluttered and obstructed environment. As previously defined, a 6D pose consists of a 3×3 rotation matrix $\boldsymbol{R}$ and a 3×1 translation $\boldsymbol{t}$.

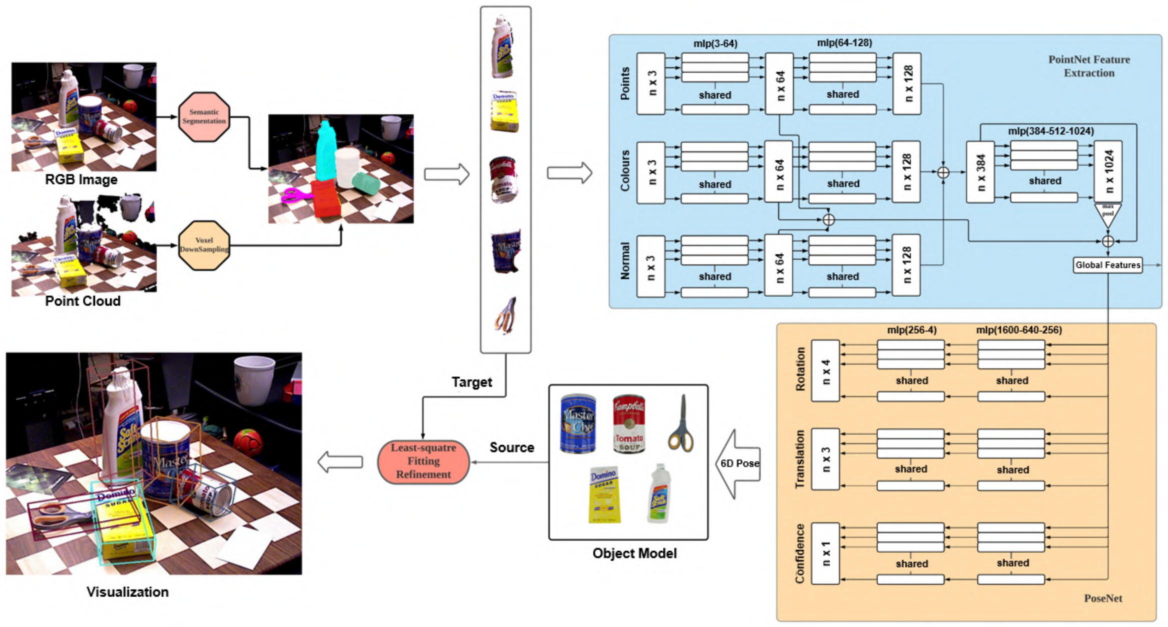


Figure 5.2: Overview of Point-6D neural network architecture. The object segmentation masks is generated from a RGB image. The feature extraction net extracts global features from a colored point cloud. The PoseNet is employed to regress translation, orientation and confidence at per-point level. A geometry-based least-square fitting algorithm is exploited to refine the 6D pose.

5.3.1 Architecture Overview

Figure 5.2 illustrates an overview of Point-6D. The system is composed by four main stages. The first step is to extract target objects mask from an RGB image, which can reduce the computational complexity of the neural network. We have introduced a widely-used semantic segmentation system, Mask-RCNN [93]. In a Densefusion pipeline [15], it performs image segmentation utilizing a segmentation network from PoseCNN [4]. The aim of this work is to develop an accurate 6D pose estimation algorithm. So, in order to conduct an efficient validation, we decided to use the same segmentation network as in the previous work [15, 4].

The second stage consists of three parallel MLP layers and one max-pooling layer, which extract and fuse feature maps with different shapes from the input point cloud. This network is inspired by PointNet, which is a generic neural network for classification and segmentation from an unstructured point cloud. This network is the backbone of Point-6D, and Figure 5.2 details the network structure in the blue areas. Given a colored point cloud segment containing n points, we extract the position, color and normal information of each point separately. The position comes from the 3D coordinates of each point. The color is normalized before being fed into the network, so the value of color is in the range (0,1). For the calculation of the normal, we adopt a hybrid scheme which has 2cm of search radius, and only considers up to 100 neighbors to compute the principal axis using a covariance analysis. After the above data preparation, the input is processed separately through a set of MLP layers with shared weights (3-64-128). Then we concatenate three $128 \times n$ features as a new $384 \times n$ feature map feeding into the next MLP layer. After max-pooling, a feature map with 1024-dimension is obtained. The final feature map $1600 \times n$ is composed by a $1024 \times n$ global features, three $128 \times n$ feature vectors and three $64 \times n$ feature vectors.

The third component is PoseNet, which is used to regress 6D pose from global features. It predicts rotation, translation and confidence independently on top of the global features. The output of the rotation branch is to predict orientation in quaternion representation. So, we add a normalization layer before output rotation. The object 6D pose output is selected by the highest confidence scoring.

The fourth stage is refinement, an iterative closet point (ICP) procedure is applied to minimise the least-squares error between the object model transformed by the network output and segmented object point cloud from the environment.

5.3.2 Learning Algorithm

The aim of the learning algorithm is to train a deep neural network which can estimate an accurate object 6D pose. As illustrated in Figure 5.3 (a), the 3D translation represents the coordinate of the object center in the camera frame. The simplest way is to regress the translation value $\mathbf{T} = (T_x, T_y, T_z)$ directly. However, this way is not the best choice since each point in an object have to regress the same values. Obviously, it is not suitable for the per-point prediction strategy. Therefore, the proposed network Point-6D estimates the point-wise 3D offset shown in Figure 5.3 (b).

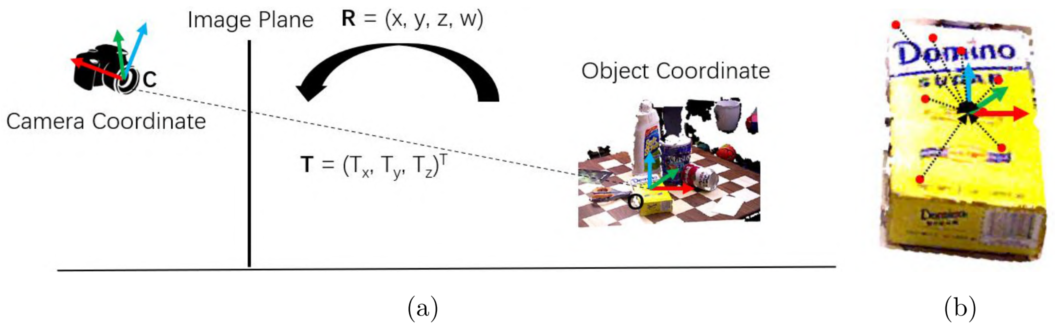


Figure 5.3: (a): Illustration of translation **t** and rotation **R** between camera coordinate and object coordinate. (b): The offset from point to the 3D center of object

There are three main ways one can represent rotation: 1) A rotation matrix; 2) axis-angle and 3) quaternion. The rotation matrix is a 3×3 matrix containing nine parameters totally, which consumes more time and memory, while axis-angle suffers from gimbal lock that will lose one degree of freedom. Therefore, we decide to utilize a normalized quaternion to represent rotation.

Apart from translation and rotation, the network also predicts a confidence score for each point. According to the confidence score, the system can determine which prediction is most likely to be the best hypothesis in the given situation. Owning to this per-point learning strategy, Point-6D is able to choose the best prediction from the visible zone of objects, which can reduce the impact of occlusion and noise.

5.3.3 Loss Function

After implementing the neural network structure, we develop a joint loss function to make the network easy to optimize. The first loss function is pose loss, which is inspired by [15, 4]. It can calculate the euclidean distance between points on an object model transformed by ground truth pose and their corresponding points transformed by prediction pose in the same model. It is defined as the following function:

$$L_{pose} = \frac{1}{N} \sum_{x \in M} \|(R(\bar{\boldsymbol{q}})x + \bar{\boldsymbol{t}}) - (R(\boldsymbol{q})x + \boldsymbol{t})\| \tag{5.1}$$

where M indicate the points set selected from an object model, N denotes the number of selected points, parameters $\boldsymbol{q}$ and $\boldsymbol{t}$ represent the predicted rotation and translation respectively, while $\bar{\boldsymbol{q}}$ and $\bar{\boldsymbol{t}}$ mean the ground truth rotation and translation and $R(\boldsymbol{q})$ is the rotation metrics converting from the quaternion $\boldsymbol{q} = (x, y, z, w)$:

$$R(\boldsymbol{q}) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2xy + 2wz & 2xz - 2wy \\ 2xy - 2wz & 1 - 2(x^2 + z^2) & 2yz + 2wx \\ 2xz + 2wy & 2yz - 2wx & 1 - 2(x^2 + y^2) \end{bmatrix} \tag{5.2}$$

However, the above loss function is specific for an asymmetric object. There will be multiple correct rotations for a symmetric object in one state. To avoid miscalculation, we instead measure the euclidean distance between points transformed by prediction pose and its closest point on the object model transformed by ground truth pose.

$$L_{pose}^{symmetric} = \frac{1}{N} \sum_{x_p \in M} \min_{x_g \in M} \|(R(\bar{\boldsymbol{q}})x_g + \bar{\boldsymbol{t}}) - (R(\boldsymbol{q})x_p + \boldsymbol{t})\| \tag{5.3}$$

So far, we have defined the loss function for the object 6D pose. As described before, apart from pose, we desire to predict the confidence score for each point. So we use a binary cross entropy loss function to measure the error between the predicted confidence c_x and the ground truth $\bar{c_x}$. Note that the ground truth of confidence for all points equates to one.

$$L_{conf} = \sum_{x \in M} c_x log\bar{c_x} + (1 - c_x) log(1 - \bar{c_x}) \tag{5.4}$$

We use the following total loss function to learning the translation, rotation and confidence jointly.

$$L = \alpha_1 L_{pose} + \alpha_2 L_{conf} \tag{5.5}$$

where α_1 and α_2 are the weights for each loss component. By assigning a property weights value, the joint loss function can improve the performance of the network. Experiments show that this joint loss function is straightforward for the network and easy to optimize.

5.3.4 Training and Implementation

The basic network architecture has been introduced in section 5.3.1. At the beginning of training, the learning rate is set to 0.001. After every ten epochs, the learning rate decays by 5% until finished at 700 epochs. As for network optimization, we choose the Adam [160] algorithm which is the most popular optimization algorithm in the field of deep learning in recent years. The Adam algorithm is suitable for optimizing the network because it is computationally efficient and has low memory requirements.

For the selection of the number of input points, we randomly choose 1000 points from the segmented object points and set $\alpha_1 = 1$ and $\alpha_2 = 0.3$ in the eqn. (5.5).

5.3.5 Iterative Close Point Refinement

ICP is an algorithm to align two point cloud set. We have introduced several ICP algorithms such as point-to-point, point-to-plane, colored ICP etc. in chapter 3. ICP and it's variants have been widely used in 6D pose refinement procedures [4, 13, 105]. The aim of ICP refinement is to minimize the point distance between segmented object points and transformed object model points. Therefore, we employ the point-to-point ICP as post-processing.

Given the two point sets, the segmented object points $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_i\}$ and the transformed object model points $\boldsymbol{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_j\}$, the ICP refinement is to minimize the following objective function:

$$F(\boldsymbol{T}) = \sum_{(p,q)\in\kappa} \|\boldsymbol{p} - (R\boldsymbol{q} + t)\|^2 \tag{5.6}$$

where κ is the correspondence set between points set $\boldsymbol{P}$ and $\boldsymbol{Q}$. Specifically, we calculate the distance from each point in $\boldsymbol{P}$ to each point in $\boldsymbol{Q}$, and return the corresponding points in $\boldsymbol{Q}$ with the smallest distance that is less than the threshold. Obviously, this retrieval distance calculation is time-consuming especially for dense point clouds. Therefore, we adopt a voxel downsampling filter as a pre-processing step to spare point cloud. Afterwards, compute the center of mass for two point cloud, and then a Singular Value Decomposition (SVD) [150] is used to update a new rotation matrix $\mathbf{R}$, while the new translation $\mathbf{t}$ is obtained by computing the difference between the centroid of $\boldsymbol{P}$ and the centroid of $\boldsymbol{Q}$ transformed by the new $\mathbf{R}$. This procedure is similar with the PnP solution steps (from eqn. (4.7) to eqn. (4.14)) introduced in section 4.3.3.

Algorithm 4 Point-to-point ICP algorithm

```
Input: Target point cloud $\boldsymbol{P}$; Source point cloud $\boldsymbol{Q}$; Initial transformation $\boldsymbol{T}^0$;
Output: Transformation $\boldsymbol{T}$ minimal the objective function 5.6
1: while not converged do
2:     Downsample for $\boldsymbol{P}$ and $\boldsymbol{Q}$.
3:     Find correspondence set $\kappa = \{(p, q)\}$ for $\boldsymbol{P}$ and $\boldsymbol{Q}$.
4:     Calculate the distance between center of mass.
5:     Compute $\mathbf{R}$ and $\mathbf{t}$ via SVD.
6:     Update transformation $\mathbf{T}$.
```

The above step is one refinement step. To get the process on needs to repeat the refinement until the change in the mean-square error is below the preset threshold. One can dot-multiply the network output pose and refinement pose to obtain the final result.

5.4 Experiments

This chapter has proposed the Point-6D system (defined in section 5.2) for achieving high accurate object 6D pose estimation. This section evaluates the efficacy and gives evidence that Point-6D outperforms the state-of-art methods. Similar to the last section, we evaluate Point-6D in two datasets; one is a single object dataset, and another is a multiple objects dataset. For single object evaluation, we continue to employ LineMOD [3]. While in multiple object experiments, we use the YCB-Video dataset [4] instead of an Occluded-LineMOD dataset because the YCB-Video contains more objects and is adopted by most RGB-D based 6D pose evaluations. In experiments, we compare Point-6D (shown in section 5.2) with the state-of-the-art 6D pose estimation algorithm.

5.4.1 Datasets

YCB-Video Dataset. The YCB-Video dataset is a widely-used dataset for 6D object pose estimation. It was proposed by Xiang et al. [4] to evaluate the performance of PoseCNN. The YCB-Video dataset contains accurate 6D poses of 21 objects with various shapes and textures. Those data are generated from 92 RGB-D videos with a total of 133827 frames captured by an Asus Xtion Pro Live RGB-D camera in fast-cropping mode, and they also provide varying lighting conditions and occlusion, making detection changeable. Figure 5.4 shows a set of the dataset using a real image. What is more, there is also a lot of synthetic data available for YCB objects. We follow the same training setting of previous works [15, 4] to split the dataset into 80 videos for training and select 2949 key frames from the remaining 12 videos for evaluation.

LineMOD Dataset. The LineMOD dataset contains 13 textureless objects that we have introduced in the last chapter. We also generate the colored point cloud for them as illustrated in Figure 5.5. We follow the same training and testing set as previous work [161, 162, 15], that is, 15% data for training and the remaining 85% for testing.

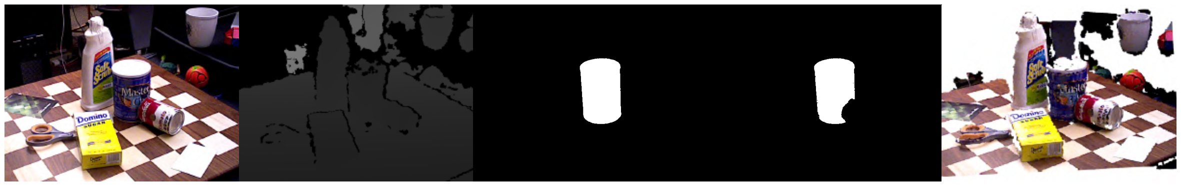


Figure 5.4: A standard YCB-Vedio dataset. Note that only one object mask is posted, and the remaining four object masks are not shown here. The point cloud is not included in the original dataset; we generate it additionally

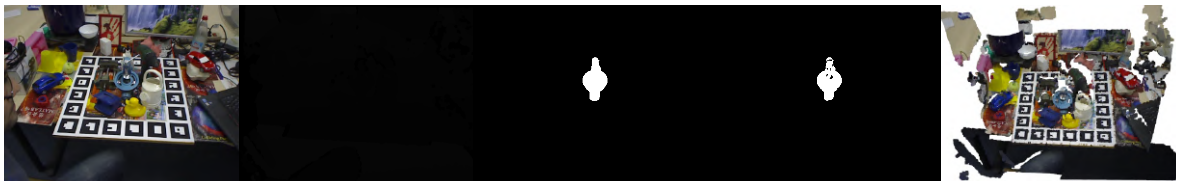


Figure 5.5: A standard LineMOD dataset.

5.4.2 Metrics

To conduct experiments with uniform metrics, we follow prior work to adopt the average distance (ADD) metric and average distance for symmetric objects (ADD-S) metric. The average distance calculates the mean of pairwise distances between object vertices calculated utilizing the estimated pose and ground truth pose:

$$ADD = \frac{1}{m} \sum_{x \in M} \underset{M}{min} \|(\boldsymbol{R}x + \boldsymbol{t}) - (\bar{\boldsymbol{R}}x + \bar{\boldsymbol{t}})\| \tag{5.7}$$

where $\boldsymbol{R}$, $\boldsymbol{t}$, $\bar{\boldsymbol{R}}$, and $\bar{\boldsymbol{t}}$ are ground true rotation, ground true translation, estimated rotation and estimated translation, respectively. M denotes the vertex set of the 3D model, and m means the number of 3D points. For symmetric objects, we use ADD-S metrics: to measure mean distance utilize closest point distance:

$$ADD - S = \frac{1}{m} \sum_{x_1 \in M} \underset{x_2 \in M}{min} \|(\boldsymbol{R}x_1 + \boldsymbol{T}) - (\bar{\boldsymbol{R}}x_2 + \bar{\boldsymbol{T}})\| \tag{5.8}$$

where x_2 is closest point of x_1 in the object model transformed by ground truth pose. Note that the per-point loss function is based on a similar principle. Besides, we exploit the area under the ADD(S) curve (AUC) in the YCB-Video evaluation, which continues the setting from [4, 15]. In the YCB-Video evaluation experiment, the maximum threshold of AUC metrics is set to 0.1m, and ADD(S) metrics is less than 2cm which is a common tolerable error for robot gripping. While in the LimeMOD evaluation experiment, the maximum threshold is 10% of the object model diameter.

5.4.3 Evaluation on the LineMOD Dataset

The LineMOD dataset contains 15 objects and we select 13 of them for the experiment. Among them, object "ape", "benchvise", "camera", "cat", "driller", "duck", "holepuncher", "iron", "lamp" and "phone" are non-symmetric, while object "eggbox" and "glue" are symmetric. Table 5.1 shows the evaluation results for the LineMOD dataset. We compare Point-6D with the state-of-art 6D pose estimation algorithm on an ADD(S) metric.

Table 5.1 presents an overall comparison between Point-6D and previous RGB-D methods. Under an ADD(S) metric, the pose estimation accuracy of Point-6D outperforms ICP post-processing approaches [163, 13, 4] by over 15%. What's more, we also outperform DeepIM [161] by 6.3% and Densefusion [15] by 0.9%. It is worth noting that the training stage of Densefusion is slow, which spends a full day to train 45 epochs for refinement. It took over ten days to complete the training of the LineMOD dataset, while, on equivalent computing, Point-6D only needed 36 hours. We visualize some sample estimation by Densefusion [15] and Point-6D in Figure 5.6.

The five sample image are randomly selected from the LineMOD test dataset. From Figure 5.6 we can observe that the DenseFusion method has significant errors in the prediction of the drill, holepuncher, and phone. While for Point-6D, although there is a rotational error in the z-axis in drill object pose estimation, the pose error of drill object is still less than ADD metric threshold. What's more, it is also worth noticing

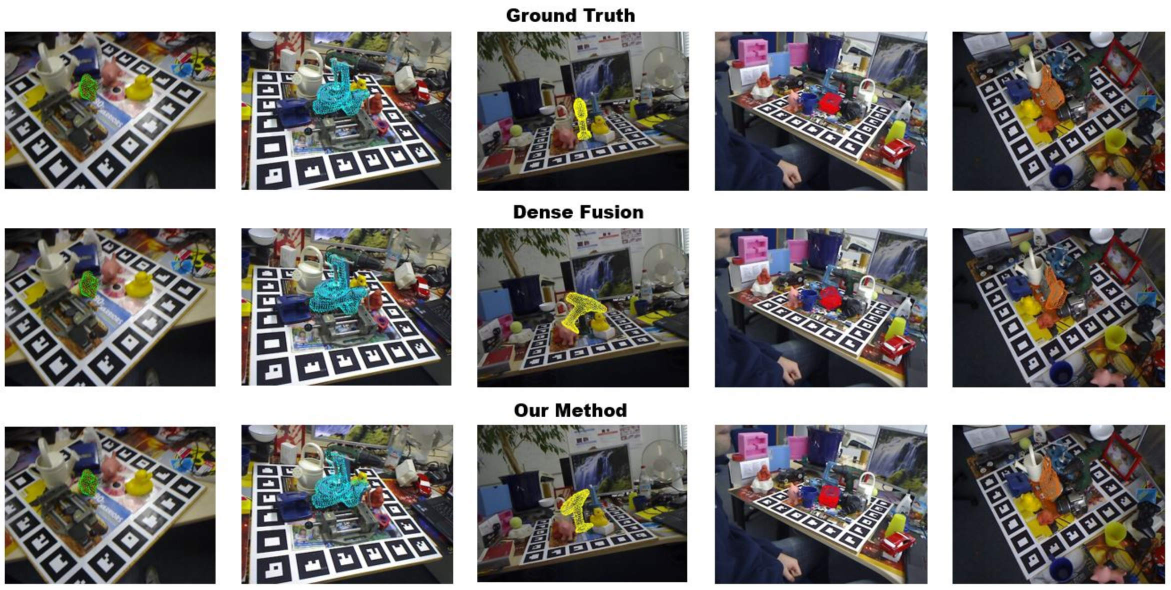


Figure 5.6: **Qualitative results on the LineMOD Dataset.** The object point cloud model is transformed with corresponds 6D pose and then projected to the 2D image via camera projection function. The first row is object projection with ground truth pose, the second row is the Dense Fusion result and the last row is Point-6D.

Table 5.1: Comparison of Point-6D with state-of-the-art work on LINEMOD data set in terms of ADD(s)-0.1 metric. We present percentages of correctly estimated pose and highlight the best result among those by **bold** numbers. Objects with **bold** name are symmetric .

Method / **Object**	Implicit + ICP	SSD-6D + ICP	PoseCNN + ICP	Deep IM	Denfusion	Point-6D
Ape	20.6	65.0	76.2	77.0	92.0	**92.1**
Benchvise	64.3	80.0	-	**97.5**	93.0	93.5
Cam	63.2	78.0	-	93.5	**94.0**	93.8
Can	76.1	86.0	87.4	**96.5**	93.0	93.6
Cat	72.0	70.0	52.2	82.1	**97.0**	95.9
Driller	41.6	73.0	90.3	**95.0**	87.0	92.1
Duck	32.4	66.0	77.7	77.7	92.0	**92.5**
Eggbox	98.6	100.0	72.2	97.1	100.0	100.0
Glue	96.4	100.0	76.7	99.4	100.0	100.0
Holepuncher	49.9	49.0	91.4	52.8	92.0	**93.5**
Iron	63.1	78.0	-	**98.3**	97.0	96.1
Lamp	91.7	73.0	-	**97.5**	95.0	96.4
Phone	71.0	79.0	-	87.8	93.0	**94.5**
Average	64.7	79.0	78.0	88.6	94.0	**94.9**

that Point-6D outperforms DenseFusion by 5.1% in drill object pose estimation. As for objects holepuncher and phone, Point-6D shows highly accurate prediction.

5.4.4 Evaluation on YCB-Video Dataset

We randomly select five samples from the test data, just as we did with the qualitative test setting in the LineMOD dataset. Figure 5.7 visualises the 6D pose result , and each color represents its corresponding object. DenseFusion has an obvious error in the first image. Objects projection of *051_largez_clamp* and *52_extra_large_clamp* are not aligned with the object in the picture in both translation and orientation. On the contrary, Point-6D predicts the accurate 6D pose for those two objects.

Table 5.2 presents the quantitative results for the YCB-Video dataset. As shown in Table 5.2, the average accuracy of the proposed 6D pose estimated method Point-6D

Table 5.2: Comparison of Point-6D with state-of-the-art methods on YCB-Video datasets in terms of ADD(s)-2cm and AUC metric. We present percentages of correctly estimated pose and highlight the best result among those by **bold** numbers. Objects with a **bold** typeface in the name are symmetric.

	PointFusion		PoseCNN		DenseFusion		Point-6D	
	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm
002_master_chef_can	90.9	99.8	95.8	100	**96.4**	100	96.1	100
003_cracker_box	80.5	62.6	92.7	91.6	95.5	99.5	**96.4**	**100**
004_sugar_box	90.4	95.4	**98.2**	100	97.5	100	96.8	100
005_tomato_soup_can	91.9	96.9	94.5	96.9	94.6	**96.9**	**95.6**	96.3
006_mustard_bottle	88.5	84.0	**98.6**	100	97.2	100	97.8	100
007_tuna_fish_can	93.8	99.8	**97.1**	100	96.6	100	96.6	100
008_pudding_box	87.5	96.7	**97.9**	100	96.5	100	97.7	100
009_gelatin_box	95.0	100.0	**98.8**	100	95.4	100	96.8	100
010_potted_meat_can	86.4	88.5	92.7	93.6	91.3	93.1	**93.6**	**94.2**
011_banana	84.7	70.5	**97.1**	99.7	96.6	100	96.7	100
019_pitcher_base	85.5	79.8	**97.8**	100	96.4	100	97.1	100
021_bleach_cleanser	81.0	60.5	96.9	99.4	95.8	100	**97.0**	100
024_bowl	75.7	24.1	81.0	54.9	88.2	**98.8**	**90.5**	98.2
025_mug	94.2	99.8	95.0	99.8	**97.1**	100	96.6	100
035_power_drill	71.5	22.8	98.2	99.6	96.0	98.7	97.2	98.6
036_wood_block	68.1	18.2	87.6	80.2	89.7	94.6	**92.5**	**95.8**
037_scissors	76.7	35.9	91.7	95.6	95.2	100	**96.2**	100
040_large_marker	87.9	80.4	97.2	99.7	**97.5**	100	95.8	100
051_large_clamp	65.9	50.0	75.2	74.9	72.9	79.2	**90.2**	**92.7**
052_extra_large_clamp	60.4	20.1	64.4	48.8	69.8	76.3	**89.7**	**90.3**
061_foam_brick	91.8	100	**97.2**	100	92.5	100	95.2	100
Average	83.9	74.1	93.0	93.2	93.1	96.8	**95.5**	**98.3**

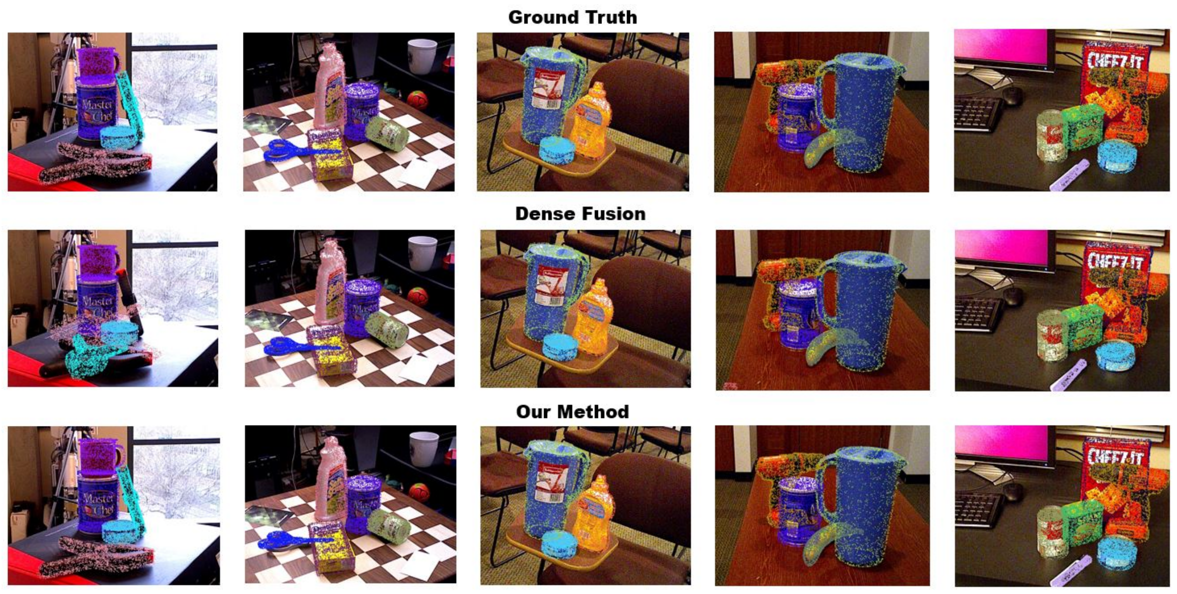


Figure 5.7: **Qualitative results on the YCB-Video Dataset.** The object point cloud model is transformed with corresponding 6D pose and then projected to the 2D image via camera projection function. The first row is the object projection with ground truth pose, the second row is the Dense Fusion result and the last row is Point-6D.

outperforms all the competitors on both AUC and ADD metrics. On the AUC metric, we surpass PointFusion [104] by 11.6%, exceeds PoseCNN [4] by 2.5% and outperforms DenseFusion [15] by 2.4%. The previous methods [104, 4, 15] did not work very well for "*051_large_clamp*" and "*052_extra_large_clamp*" object. Out method improves the performance on those two object by a large margin. The first qualitative results in Figure 5.7 gives an intuitive impression of the improvement. We also demonstrate the evaluation on an ADD metric and Point-6D also achieves the best performance on average estimation accuracy.

5.4.5 Ablation Study

In this section, we investigate the influence of different voxel downsampling values in the refinement phase for 6D pose estimation. Figure 5.8 illustrates the object point cloud with different voxel values.

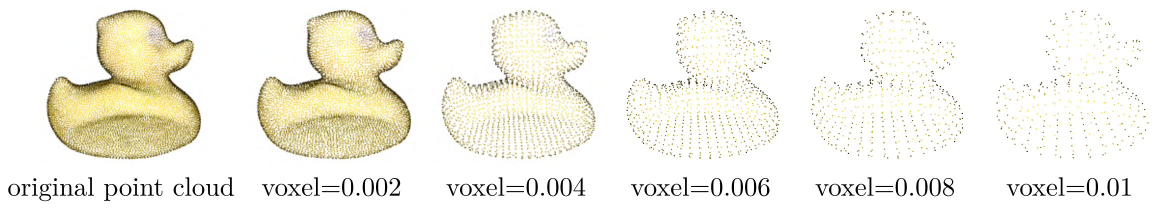
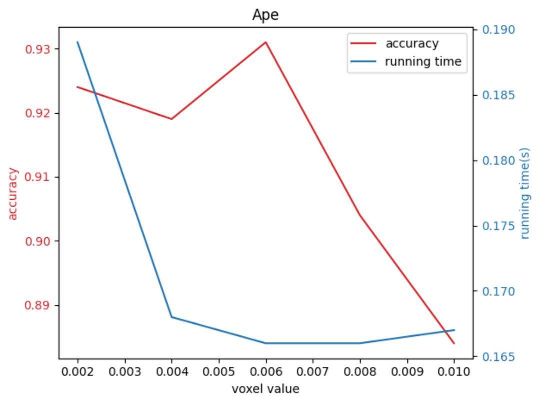


Figure 5.8: Duck point cloud model with different voxel size

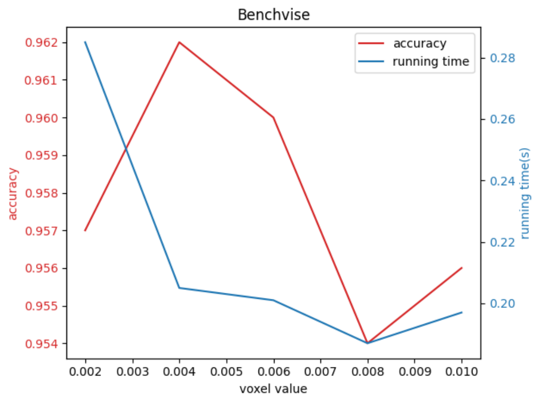
From Figure 5.8, it can be clearly seen that the object point cloud model is getting sparse as the voxel value increases. Theoretically, the running time is negatively correlated with the number of source points; as the number of points is fewer, the speed of execution is faster. However, the experimental results are not so.

We select six objects with different sizes to conduct this experiment and the results are illustrated in Figure 5.9. For objects ape, benchvise and phone, the overall trends are downward. But for the remaining three objects, the running time is not directly related to the number of points. It is possible, therefore, that the iteration number and criteria impact on the refinement running time. Even though there are fewer input points, it takes a long time if the refinement iteration cannot converge quickly. Besides, another finding is that the average time decreases as the diameter of the object decreases.

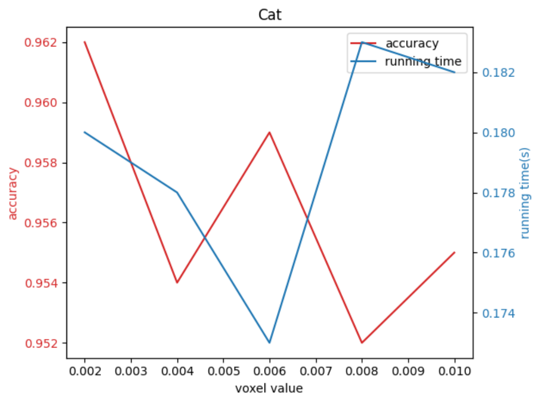
For the estimation accuracy indicated by the red line in Figure 5.8, voxel = 0.06 is a very desirable value for the post-processing. For the evaluation experiments conducted before, we set the voxel value to 0.06.



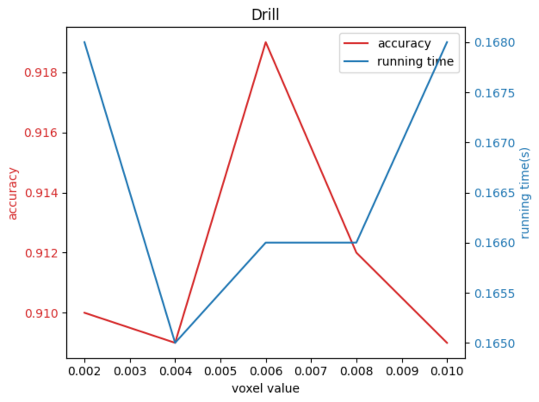
(a) Experiment on Ape



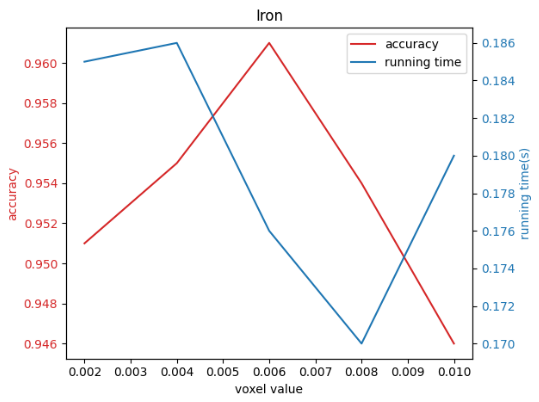
(b) Experiment on Benchvise



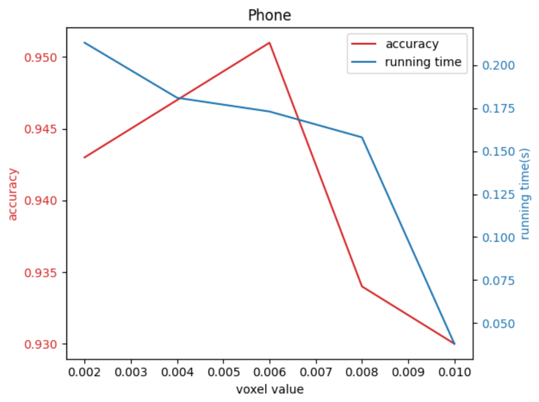
(c) Experiment on Cat



(d) Experiment on Drill



(e) Experiment on Iron



(f) Experiment on Phone

Figure 5.9: Accuracy and running time of 6D pose estimation with different down-sampling voxel value. The running time means the time consumed by one frame.

5.5 Summary

In this chapter, we proposed an object 6D pose estimation network Point-6D (that is section 5.2) shown in Figure 5.1 using a point cloud as input. The proposed deep neural network, Point-6D, can merge points, colors, and normal features from three channels to regress the 6D pose of known objects per-point level. Besides, we demonstrated the performance of Point-6D and state-of-art 6D pose estimation systems on LineMOD [3] and YCB-Video [4] datasets. For the evaluation on the LineMOD dataset, Point-6D outperforms state-of-art method PoseCNN by 16.9 %, DeepIM [161] by 6.3% and DenseFusion by 0.9% on average accuracy. For the evaluation on the YCB-Video dataset, we surpass PointFusion [104] by 11.6%, exceed PoseCNN [4] by 2.5% and outperform DenseFusion [15] by 2.4%. At the end of the chapter, we investigated the impact of different voxel downsampling values in the refinement phase for 6D pose estimation and found out the best choices for the voxel values.

Chapter 6

Conclusions

The main motivation of the project has been to use artificial intelligence to replace humans in dangerous and repetitive tasks. This thesis introduces three novel algorithmic contributions and a case study to show the power of the proposed 3D computer vision system. In the sections that follow, a summary of the key contributions is presented in Section 7.1 and potential future research directions are discussed in Section 7.2.

6.1 Summary of Contribution

Chapter 3 proposes an environment reconstruction algorithm by using a colored point cloud. Compared with current widely-used approaches, the method is distinct for the following reasons: (1) the point cloud fusion algorithm 2 takes advantage of photometric and geometric information to improve point cloud registration accuracy; (2) we employ Levenberg-Marquardt to replace a Newton-Gaussian function in the iteration step and (3) we develop a hierarchical coarse-to-fine strategy that can reduce the pose drifting issue. Furthermore, we evaluate the proposal in RGB-D Scenes and the SUN3D dataset. The former one is a small scale dataset and the latter one is a large scale indoor dataset. Apart from these, we conduct an evaluation on a real-world scene. These experiments prove the effectiveness of the proposed point cloud fusion Algorithm 2.

Chapter 4 and Chapter 5 focus on objects' 6D pose estimation. An RGB-based object 6D pose estimation algorithm is proposed by exploiting CNNs. This novel 6D pose estimation method can detect objects, segment instances and predict 6D pose simultaneously from a single RGB image without any PnP process. Besides, we introduce Contour-Alignment, an efficient algorithm for pose refinement in an RGB image. The CA refinement Algorithm 3 only needs two iterations to reduce the translation error by 70%. At the end of Chapter 4, effectiveness is evaluated on a LineMOD dataset for single object pose estimation and LineMOD-Occluded for a multiple objects pose estimation dataset. In addition, we also demonstrated how to use synthetic data to train custom objects.

Chapter 4 mainly considers the scenarios where depth information is not available. Chapter 5 proposes an object 6D pose estimation algorithm using a point cloud as input. The novel deep neural network, Point-6D, can merge points, colours, and normal features from three channels to regress the 6D pose of known objects per-point level. Besides, the performance of Point-6D is demonstrated in comparison with state-of-art 6D pose estimation system on LineMOD [3] and YCB-Video [4] datasets. For the evaluation in the LineMOD dataset, Point-6D outperforms the state-of-art method PoseCNN by 16.9 %, DeepIM [161] by 6.3% and DenseFusion by 0.9% on average accuracy. At the end of the chapter, we investigate the impact of different voxel down-sampling values in the refinement phase for 6D pose estimation and present the best choices for voxel values.

6.2 Future Research Directions

Although a lot of work has been completed in this thesis, we are just scratching the tip of the iceberg in the field of 3D robotic vision. In future studies, there are a number of numerically fascinating directions that are worth investigation.

Self-supervised learning: The learning-based 6D pose estimation methods in this thesis are supervised learning, which rely on ground truth data to learn. In most scenarios, it is difficult to acquire accurate 3D ground truth data. Furthermore, the annotation of 6D pose is complicated and laborious. In this thesis, a synthetic dataset is introduced, which alleviates the above limitations slightly. But the migration between synthetic data and real data makes it a challenge. The trained models having a good performance in synthetic data does not necessarily mean that they will have the same good performance on real data. Some recent works Deng et al. [27], Wang et al. [28] have considered utilizing self-supervision in 6D pose estimation to create robot annotated data autonomously. However, these require a precise target object model as a template or source of the synthetic training data. Therefore, an exciting direction for research is to let the robot label object 6D pose according to 2D clues.

High level robotic autonomy: The robotic system in this thesis is a semi autonomous system, whereby the robot can complete tasks according to the operator's commands. Assuming that the robot has a working environment model and objects and their location in the environment; it is an interesting research direction to make the robot decide what action it should perform rather than relying on direction from the operator. A possible approach for future autonomy may be provided by the sEnglish based robot reasoning scripting system as presented at `www.transparentrobots.org` and in [164] using natural language programming (NLP) [165].

Deep reinforcement learning for robotic manipulation: Recently, reinforcement learning [166] combined with convolutional neural network has applied successfully in learning policies in the field of robotic manipulation. Due to the use of reinforcement learning, robot is able to directly learn dexterous manipulation from raw images. In the field of robotics, reinforcement learning normally represents continuous high-dimensional action and state space [167]. Levine et al. [168] proposed a novel reinforcement learning with a CNN grasp predictor to learn objective, and a servoing mechanism to utilize this predictor to optimize grasp performance. The demonstration shows that the robot can constantly adjust its grasp pose with the use of continuous feedback, and successfully grasp in the clutter on a wide range of objects. However, to generate enough data, Levine et al. [168] used up to 9 of robot collecting over

900,000 grasp attempts, which is quite time-consuming. To improve sample efficiency, Wulfmeier et al. [169] developed a reinforcement learning system with an alignment rewards that encourage both agents in simulated and real robots to have similar distributions over visited states. So an exciting research for future work would to how to quickly implement data collection across many deployed robots engaged in both real world and simulation environment. In addition, imitation learning [170] with a mentor provides demonstrations, meta learning [171] that can train a model on a variety of learning tasks, and inverse reinforcement learning [172] that benefits designing reasonable reward functions are all worth investing deeply in the future.

Appendices

Appendix A

Methods for Non-Linear Least Squares Problems

In the field of 3D vision, whether in creating 3D-3D correspondence that is involved in Chapters 3 and 5 or 3D-2D correspondence that is used in Chapter 4 each of these require the solution of non-linear least squares problems. This section presents two classical methods for solving non-linear least squares optimisation: the Gauss-Newton method and the Levenberg-Marquardt method.

A.1 The Gauss-Newton Method

For a non-linear least squares problem:

$$\boldsymbol{x} = argmin_x F(\boldsymbol{x}) \tag{A.1}$$

$$F(\boldsymbol{x}) = \frac{1}{2}\|f(\boldsymbol{x})^2\| \tag{A.2}$$

The Gauss-Newton method uses the Taylor expansion to obtain a linear approximation:

$$f(\boldsymbol{x} + \boldsymbol{\Delta x}) \simeq \boldsymbol{l}(\boldsymbol{\Delta x}) \equiv f(\boldsymbol{x}) + J(\boldsymbol{x})\boldsymbol{\Delta x} \tag{A.3}$$

where J is the Jacobian matrix that includes the first partial derivatives of the function:

$$J = [\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_n}] \tag{A.4}$$

Inserting eqn. (A.2) into eqn. (A.3) we derive that:

$$\begin{aligned} F(\boldsymbol{x} + \boldsymbol{\Delta x}) \simeq \boldsymbol{L}(\boldsymbol{\Delta x}) &\equiv \frac{1}{2}\boldsymbol{l}(\boldsymbol{\Delta x})^T \boldsymbol{l}(\boldsymbol{\Delta x}) \\ &= \frac{1}{2} f(\boldsymbol{x})^T f(\boldsymbol{x}) + \boldsymbol{\Delta x}^T J(\boldsymbol{x})^T f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\Delta x}^T J(\boldsymbol{x})^T J(\boldsymbol{x}) \boldsymbol{\Delta x} \\ &= F(\boldsymbol{x}) + \boldsymbol{\Delta x}^T J(\boldsymbol{x})^T f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\Delta x}^T J(\boldsymbol{x})^T J(\boldsymbol{x}) \boldsymbol{\Delta x} \end{aligned} \tag{A.5}$$

Take the derivative of the above formula and set the derivative to 0, we obtain:

$$J(\boldsymbol{x})^T J(\boldsymbol{x}) \boldsymbol{\Delta x} = -J(\boldsymbol{x})^T f(\boldsymbol{x}) \tag{A.6}$$

Then transmitting eqn. (A.6) can solve the increment value Δx:

$$\boldsymbol{\Delta x} = -(J(\boldsymbol{x})^T J(\boldsymbol{x}))^{-1} J(\boldsymbol{x})^T f(\boldsymbol{x}) \tag{A.7}$$

Updating of $\boldsymbol{x}$ by

$$\boldsymbol{x} := \boldsymbol{x} + \boldsymbol{\Delta x} \tag{A.8}$$

as above can be carried on until a terminating criterion is met, such as no significant reduction in the value of $F(\boldsymbol{x})$.

A.2 The Levenberg-Marquardt Method

Levenberg and later Marquardt proposed the Levenberg-Marquardt(LM) algorithm on the basis of Gauss-Newton method to solve non-linear least squares problems. The LM algorithm adds a trust region into the Gauss-Newton method to constrain the range of Δx. Within the trust region, we consider the approximation to be valid; outside the trust region the approximation can be unreliable. In the LM algorithm, eqn. (A.6) is modified to :

$$(J(\boldsymbol{x})^T J(\boldsymbol{x}) + \mu \boldsymbol{I})\boldsymbol{\Delta x} = -J(\boldsymbol{x})^T f(\boldsymbol{x}) \tag{A.9}$$

where I is the identity matrix and μ denotes damping scalar such that $\mu \geq 0$. For the choice of the μ the principles is followed that the initial value of μ should depends on the size of the elements in $\boldsymbol{A} = J(\boldsymbol{x_0})^T J(\boldsymbol{x_0})$, hence:

$$\mu_0 = max_i\{a_{ii}\} \tag{A.10}$$

where a_{ii} represent the diagonal elements of $\boldsymbol{A}$ and the rest of μ is obtained by a widely used strategy [173] recursively:

$$\begin{aligned}
&\textbf{if } \varphi > 0 \\
&\qquad \mu := \mu * max\{\frac{1}{3}, 1 - (2\varphi - 1)^3\}; \\
&\textbf{else} \\
&\qquad \mu := \mu * \nu; \quad \nu := \nu * 2
\end{aligned} \tag{A.11}$$

where the initial value of factor ν is 2 and φ is the gain ratio that is obtained with the value of μ from the previous step by:

$$\varphi = \frac{F(\boldsymbol{x}) - F(\boldsymbol{x} + \boldsymbol{\Delta x})}{|L(\boldsymbol{0}) - L(\boldsymbol{\Delta x})|} \tag{A.12}$$

Here the denominator is derived from eqn. (A.5) and eqn. (A.9):

$$\begin{aligned}
L(\boldsymbol{0}) - L(\boldsymbol{\Delta x}) &= F(\boldsymbol{x}) - (F(\boldsymbol{x}) + \boldsymbol{\Delta x}^T J(\boldsymbol{x})^T f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{\Delta x}^T J(\boldsymbol{x})^T J(\boldsymbol{x})\boldsymbol{\Delta x}) \\
&= -\boldsymbol{\Delta x}^T J(\boldsymbol{x})^T f(\boldsymbol{x}) - \frac{1}{2}\boldsymbol{\Delta x}^T J(\boldsymbol{x})^T J(\boldsymbol{x})\boldsymbol{\Delta x}) \\
&= -\frac{1}{2}\boldsymbol{\Delta x}^T[2J(\boldsymbol{x})^T f(\boldsymbol{x}) + (J(\boldsymbol{x})^T J(\boldsymbol{x}) + \mu\boldsymbol{I} - \mu\boldsymbol{I})\boldsymbol{\Delta x}] \\
&= -\frac{1}{2}\boldsymbol{\Delta x}^T[2J(\boldsymbol{x})^T f(\boldsymbol{x}) + (-J(\boldsymbol{x})^T f(\boldsymbol{x}) - \mu\boldsymbol{I}\boldsymbol{\Delta x}] \\
&= -\frac{1}{2}\boldsymbol{\Delta x}^T(J(\boldsymbol{x})^T f(\boldsymbol{x}) - \mu\boldsymbol{\Delta x})
\end{aligned} \tag{A.13}$$

Inserting μ into the eqn. (A.9) gets value $\boldsymbol{\Delta x}$:

$$\boldsymbol{\Delta x} = -(J(\boldsymbol{x})^T J(\boldsymbol{x}) + \mu I)^{-1} J(\boldsymbol{x})^T f(\boldsymbol{x}) \tag{A.14}$$

For a fixed $\boldsymbol{x}$ the μ and $\boldsymbol{\Delta x}$ are to be updated until the convergence criterion is reached. Generally, the convergence criterion includes maximum iteration, relative fitness and relative RMSE.

The $\boldsymbol{x}$ itself is updated by

$$\boldsymbol{x} := \boldsymbol{x} + \boldsymbol{\Delta x} \tag{A.15}$$

until we meet the terminating criterion is met in terms of the reduction of $F(\boldsymbol{x})$.

A.3 Discussion

From eqn. (A.9) we can notice that Gauss-Newton requires that the matrix $J(\boldsymbol{x})^T J(\boldsymbol{x})$ is positive definite and invertible. However, in the computation, if a matrix $J(\boldsymbol{x})$ is a positive semi-definite matrix that means the matrix $J(\boldsymbol{x})^T J(\boldsymbol{x})$ may be a singular matrix. Consequently, this can result in increment $\boldsymbol{\Delta x}$ being unstable and lead to non-convergence of the iteration.

While in the LM algorithm, a trust region is added to the increment $\boldsymbol{\Delta x}$. The damping parameter μ is the trust region radius, and its effects are:

1. When the damping parameter μ is small, $J(\boldsymbol{x})^T J(\boldsymbol{x})$ dominates, indicating that the quadratic model can get convergence. In this situation, the LM algorithm is closer to the Gauss-Newton method.

2. When the damping parameter μ is large, $\boldsymbol{\mu} I$ dominates, indicating that the $J(\boldsymbol{x})^T J(\boldsymbol{x})$ maybe a singular Matrix. In this situation, we actually use the gradient descent method to solve the non-linear least squares problem, which can avoid non-convergence of the iteration.

Appendix B

Case Study: An Advanced Teleoperation System with Intelligent Vision System

In the previous chapters a point cloud environment reconstruction system, an image-based 6D pose estimation method, and a point cloud-based 6D pose estimation method Point-6D were introduced. In this chapter those techniques are applied to a teleoperation system to help operators improve their work efficiency and reduce their workload. The 3D environment reconstruction Algorithm 2 can provide a high-quality 3D point cloud model for unknown scenes that can improve their scene understanding. The object detection system can improve the autonomous level of the teleoperation system that reduces the workload of operators.

B.1 Introduction

Recently, there has been an increasing number of teleoperated robotic systems deployed in unstructured and hazardous environments, such as the nuclear industry, search and rescue, manufacturing and the international space station, etc. [174]. However, current teleoperations are mostly based on multi-screen displays [175, 176], which means operators need to view the multiple video-stream cameras placed in the remote environment and also monitor the robot's state in physical space at any point in time. As a result, operators have to shift their concentration between monitoring the remote camera's video feed and monitoring the robot states constantly, which may lead to fatigue and stress for operators. Apart from this, the operator's 3D perception of the remote environment during teleoperation depends on a 2D projection of a 3D environment. This process can be time-consuming, inefficient and less productive.

An advanced 3D visualisation system of the remote environment requires explicit 3D reconstruction scenes. However, the current environment reconstruction technologies cannot meet the demand, especially for the small-scale texture-less scenes. Offline methods often need hours to process, while online approaches have a severe weakness which is their reliance on object surface texture to compute the corresponding relations between different frames.

Furthermore, an excellent tele-operation system should help operators improve their work efficiency. The majority of state-of-art systems still excessively depend on human intelligence. With the development of 6D pose estimation technology, an increasing number of research projects achieved semantic grasping in a cluttered environment. Despite this fact, creating vision algorithms for 6D pose estimation in tele-operation system is still a challenge. Template-based algorithms (e.g., LineMOD [8], PWP3D [177]) depend on the intact and high-quality 3D object model to extract texture features of the object and compare it with scenes. In current industrial applications, all objects manipulated by robots are made of similar material and cannot easily be differentiated by features such as texture or colour. Deep learning based-methods (e.g., DOPE [178]) can overcome this slightly, but those systems inevitably require long training times and enormous computing resources.

To bridge such contradictory technical challenges, we developed a novel graphical user interface (GUI) for teleoperation based on previous proposed works. A robot can first scan the surrounding environment and generate a high-quality 3D realistic model in this system. Second, the object lists are visualised based on the detection history and operator's guidance from the visual recognition system. Finally, operators can select the target object which needs to be grasped by robots. Through a proposed graphical user interface, the operator can interact with the environment easily. Most importantly, the developed system can run on both unknown objects and scenes first "seen".

The main contributions of this chapter are as follows:

- Developed a novel 3D robotic vision system for teleoperation to reduce workload and improve efficiency.
- Conducted a quantitative experiment in a simulated nuclear waste disposal scenario.

Note that most teleoperation in the past required visualisation using 2D interfaces [179, 180, 181], 3D point clouds [182] or object surface meshes [183].

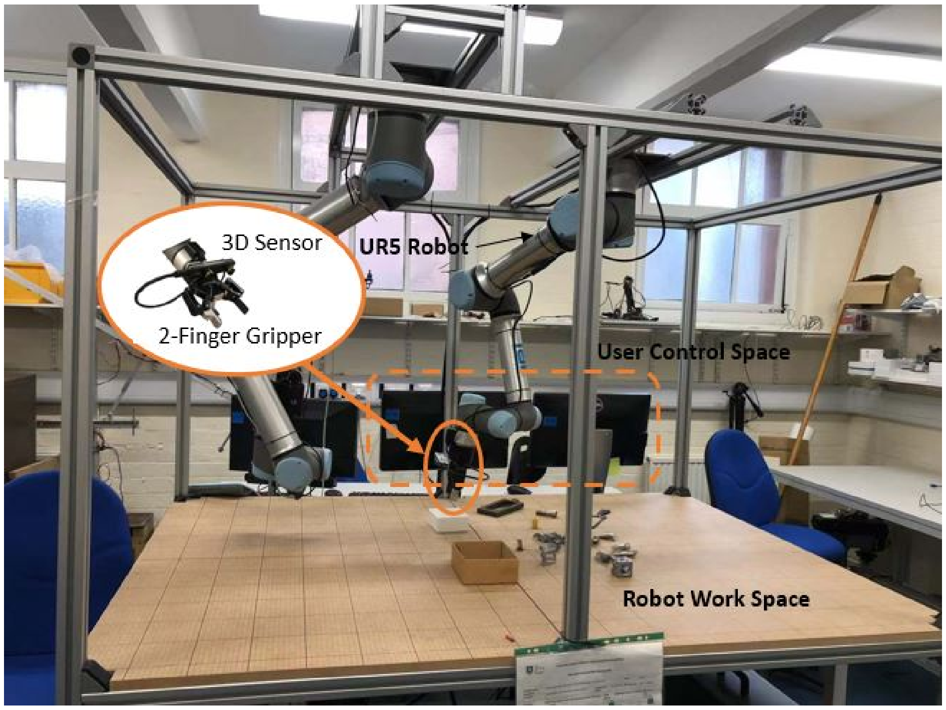
The remainder of this chapter is organised as follows: section 2 gives an overview of the related literature; section 3 details the pipeline of the 3D environment reconstruction system. Section 4 introduces the human-supervised semi-autonomous system. Section 5 presents the experiments and results. Finally, conclusions are drawn in Section 6.

B.2 Intelligent 3D Robotic Vision System

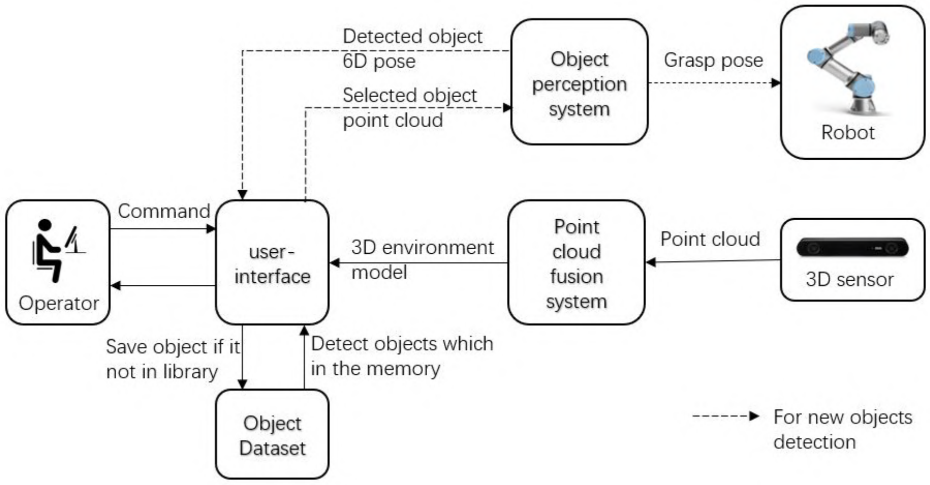
Figure B.1 conceptually depicts the architecture of the intelligent teleoperation system. In this section, we detail this teleoperation system from both hardware and software aspects.

B.2.1 Hardware Configuration

UR5 Robot Arm. The Universal Robot UR5 robot is a lightweight robotic arm with 6 Degrees of Freedom (DoF), which can engage in varying tasks with reasonable flexi-



(a) Robot configuration



(b) Block diagram of the teleoperation system

Figure B.1: System Architecture

bility. It consists of six separate joints with the ranges of +/- 360° and the maximum working radius up to 850mm. The robot is hanging on a work cell that extends the range of movements and facilitates an environment scan.

ROBOTIQ 2F-85 Gripper. ROBOTIQ 2F-85 Gripper has two articulated fingers with two joints each (two phalanxes per finger) which are shown in Figure B.1. The stroke of the gripper is 85mm while its form-fit grip payload achieves 5 kg. The ROBOTIQ 2F-85 Gripper can automatically adapt to the shape of the target object grasped.

Zed Mini Camera. For the data acquisition, we adopt a zed-mini stereo camera fixed on the robot end-effector. Zed-mini is a cost-effective dual lens 3D sensor (stereo camera) supporting up 2.2k video stream. It can sense depth from 0.1 meters to 12 meters with high accuracy and low energy. Therefore, the Zed-mini camera is widely used in virtual reality, SLAM, 3D object detection and other 3D vision applications.

Host Computer. All the data is processed by an Alienware R11 desktop. The graphics processing unit (GPU) of the computer is RTX 3080 which has 10 GB memory and operates at a frequency of 1440 MHz. The central processing unit (CPU) uses a Inter Core i9-10900 featuring 10 cores with 20 threads. The maximum frequency of this processor is up to 5.3 GHz. Due to this high specification computer, the system is able to process plenty of data and render the point cloud as a 3D model at a speed that is suitable for some industrial applications.

The above components are the hardware configuration. It is also worth mentioning that the size of the robotic work cell is 180 cm $\times$ 130 cm $\times$ 120 cm, which is suitable for the robot's movements to handle objects on the workbench.

B.2.2 Software Configuration

The robotic software system is built around the ROS (Robot Operation System) that can streamline system integration and data exchange. ROS is an open-source framework providing various tools, libraries, and conventions for robotic systems. The advantage of using ROS is the hardware abstraction and low-level control of the robot without the user knowing all the details of the robot. Furthermore, the ROS-based development enables the system to be deployed in industrial sites easily.

UR5 Robot Package. We exploit *ROS-Industrial Universal Robot meta-package* to provide a stable and sustainable interface between UR5 robot and ROS. This package includes essential nodes for communication with the UR robot controller, the unified robotic description format (URDF) models and the related MoveIt ROS packages for various models of UR robots.

ROBOTIQ 2F-85 Gripper Package. The drive of the ROBOTIQ 2F gripper is from the *ROS-Industrial Robotiq meta-package.* This package provides URDF models and a robotiq_2f_action_server node that can control the position, speed and force of the gripper.

Zed-mini Drive. The zed-mini drive is from *ZED SDK on stereolabs.com.* To work with ROS together, we use the *zed_ros_wrapper package.* This package can switch the camera to left or right, rectify/unrectify images, create depth map, create coloured dense 3D point cloud. It can also output odometer trajectory, containing position and orientation estimates of the camera based on measurements by the IMU.

MoveIt Package. The MoveIt Motion Planning Framework is an open source robotics manipulation platform integrated with many useful packages including motion planning, manipulation, inverse kinematics, control, 3D perception, collision checking, etc. With the help of MoveIt, the robot can generate a high-degree of freedom trajectories automatically for given gripper and object pose. After a plan is tested in virtual reality, the computed joint trajectories can be transmitted to hardware controllers to be executed.

B.2.3 Graphical User Interface

To improve operator experience and efficiency, a concise GUI has been developed as shown in Figure B.2. The operator needs to use mouse clicks to send high-level commands to the robot. The next section details the function and implementation of each component of this interface.

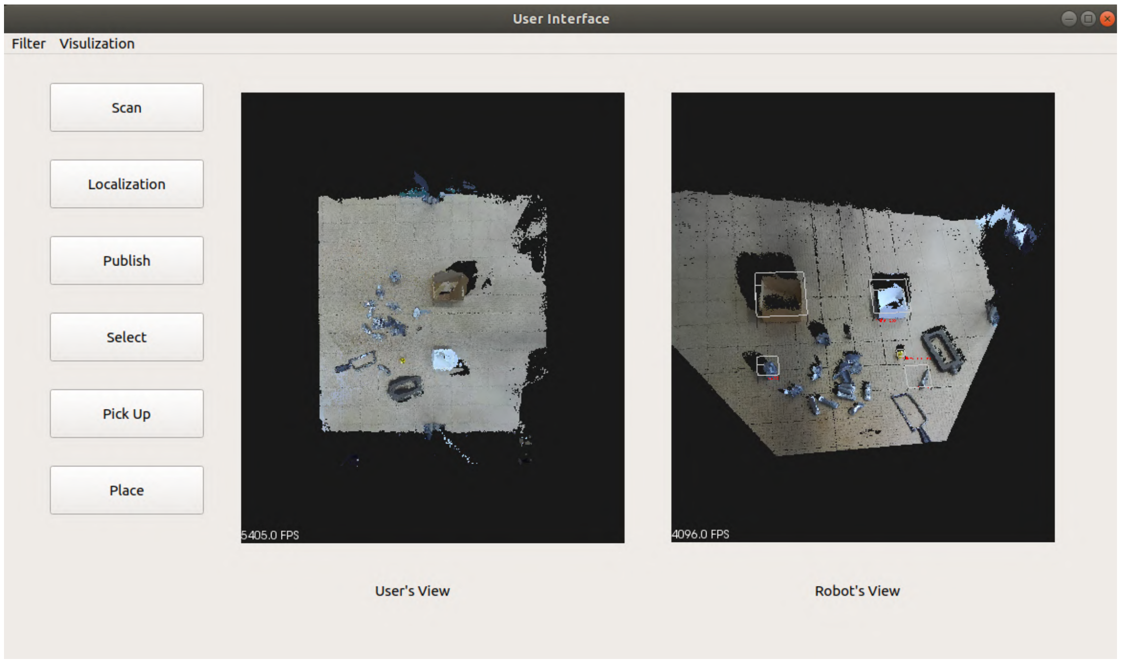


Figure B.2: The graphical user interface

Scan. When the scan button is clicked, the robot camera starts moving along a preset scan path of camera poses as the arm moves it around. For a circular trajectory of the preset path, the operator needs to select eight sample poses with 45° increments, as shown in 8 images in Figure B.3. During the scanning process the camera's software generates coloured point clouds at each pose. The captured coloured point clouds are merged using the fusion algorithm (Algorithm 2) as proposed in Chapter 3. More specifically, the point cloud is transformed from camera coordinates to customised world coordinates, that is, linked to the centre of the workbench. Utilizing the point cloud fusion system outlined in Figure 3.3, the eight point clouds that are captured from multi-view are merged to one scene model. The scene model is published via a ROS topic as *sensor_msgs/PointCloud2* message. The GUI and RVIZ (a ROS visualisation program) can display the point cloud model together with thee RVIZ Moveit virtual reality environment as based on Gazebo (a VR simulator associated with ROS) as shown in Figure B.4.

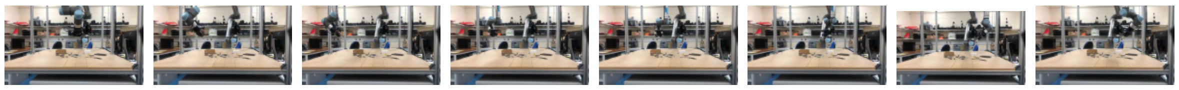


Figure B.3: Eight sampling points for robot arm

Localization of objects. The localization algorithm provides the required object's position, which is needed for grasping. In this case study, two situations are considered : (1) known objects (2) unknown objects. For known objects, we can detect objects via a neural network as presented in Chapter 5. In this work, we assume that the diameter of the objects to be grasped is less than the maximum stroke of the gripper. The adopted gripper can pick up objects without calculating grasp orientation. Therefore, we convert the problem to an instance segmentation task that extracts object position information only.

Select. The challenge of supervised learning is to label ground truth data. After manually labeling the point cloud data, we can train a network to predict object points in the scene and compute the centroid of the object. This component is designed for tasks under unknown scenes. The neural networks will fail when it comes to a new environment. Therefore, the object position needs to be obtained with the help of human intelligence. In the GUI, operators can select objects via the left window that displays the whole point cloud model. This procedure can be done easily through mouse clicking. After entering an object name, we can obtain an object's grasping location. Figure B.5 illustrates the processing of selecting object.

Publish in ROS. All the detected objects are published using a ROS topic with message type of *geometry_msgs/PoseStamped*. Toward this end, we can achieve grasping of objects by subscribing a corresponding pose topic, which can streamline the data exchange.

Pick Up. When users click the “Pick Up” button, there is a new window pop-up that lists all the detected objects as shown in Figure B.6 (a). The user can send the command to the robot controller via "Select" to indicate which object needs to be

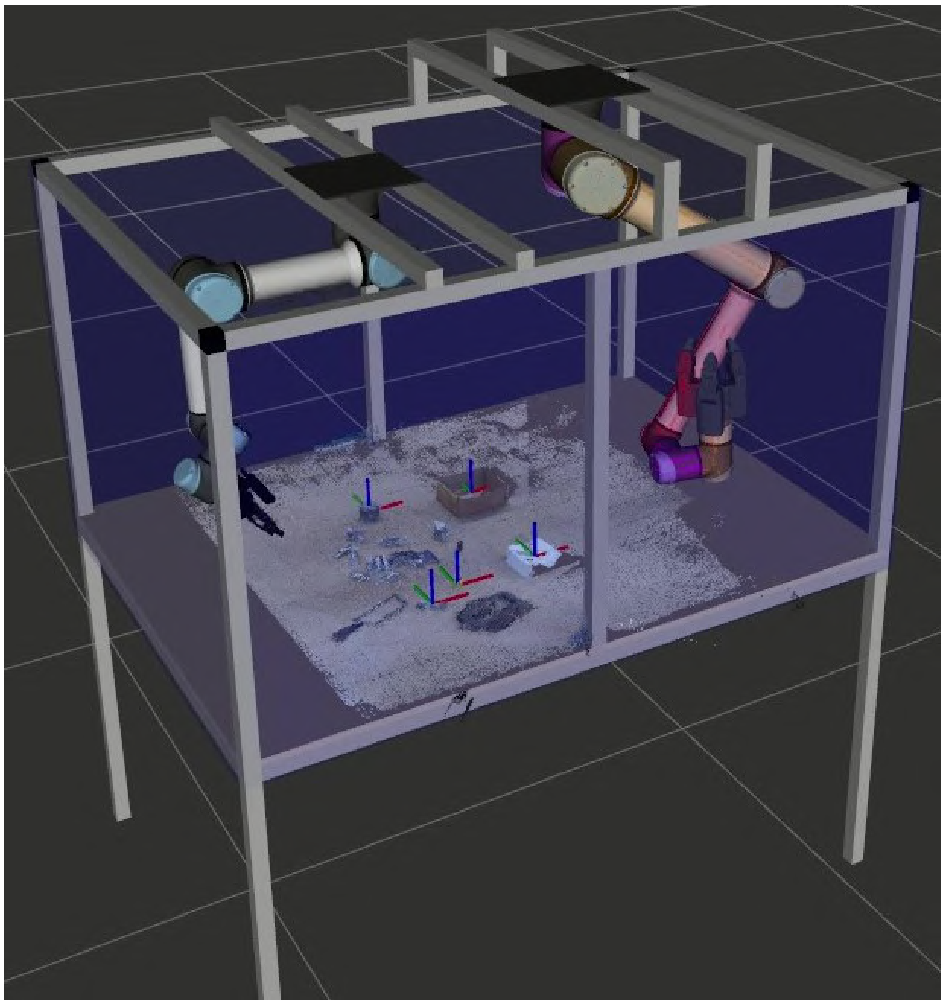
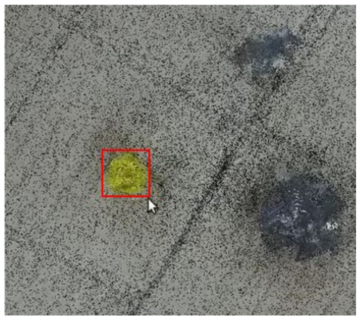
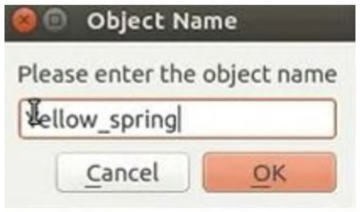


Figure B.4: Environment modeling

picked up.

Place. When users click the “Place” button, there is a new window pop-up as well (illustrated in Figure B.6 (b)). Apart from the list of objects, an option is provide for users to select where they want to place an object.


(a) Selecting target object


Object Name

Please enter the object name

yellow_spring

Cancel OK

(b) Entering object name

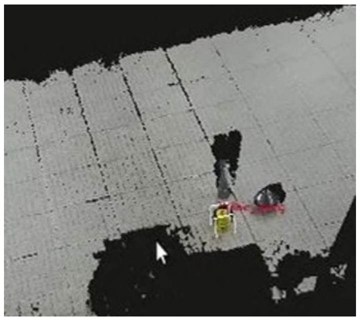
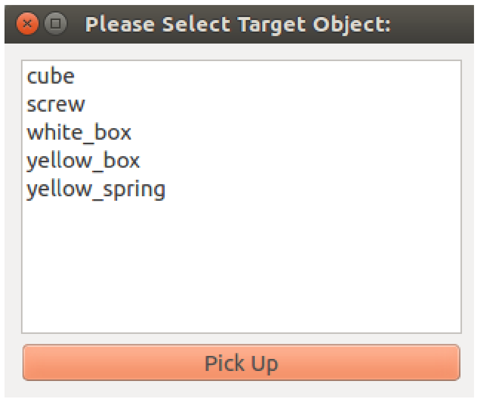

(c) Localizing object

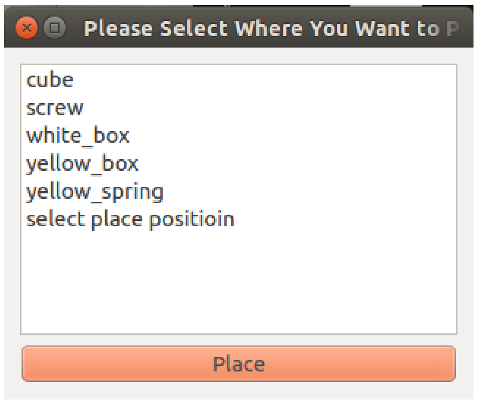
Figure B.5: Operator select target object.


Please Select Target Object:

cube
screw
white_box
yellow_box
yellow_spring

Pick Up

(a) Selecting target object


Please Select Where You Want to P

cube
screw
white_box
yellow_box
yellow_spring
select place positioin

Place

(b) Entering object name

Figure B.6: Pick-up and place objects list

B.3 Experiment

The proposed system has been applied to a laboratory nuclear waste handling scenario shown in Figure B.7 to verify the effectiveness of the system[1]. More specifically, the task of this experiment is to pick up the cube object and place in the yellow box, as well as pick up the yellow spring and place in the white box.

In the experiments we have conducted, we compared two different teleoperation modes, one is using the UR5e robot controller panel that is manipulated manually.

[1]The laboratory of the setup was provided under the EPSRC RAIN (Robotic AI in Nuclear) project lead by Professor Sandor M Veres at Sheffield University.

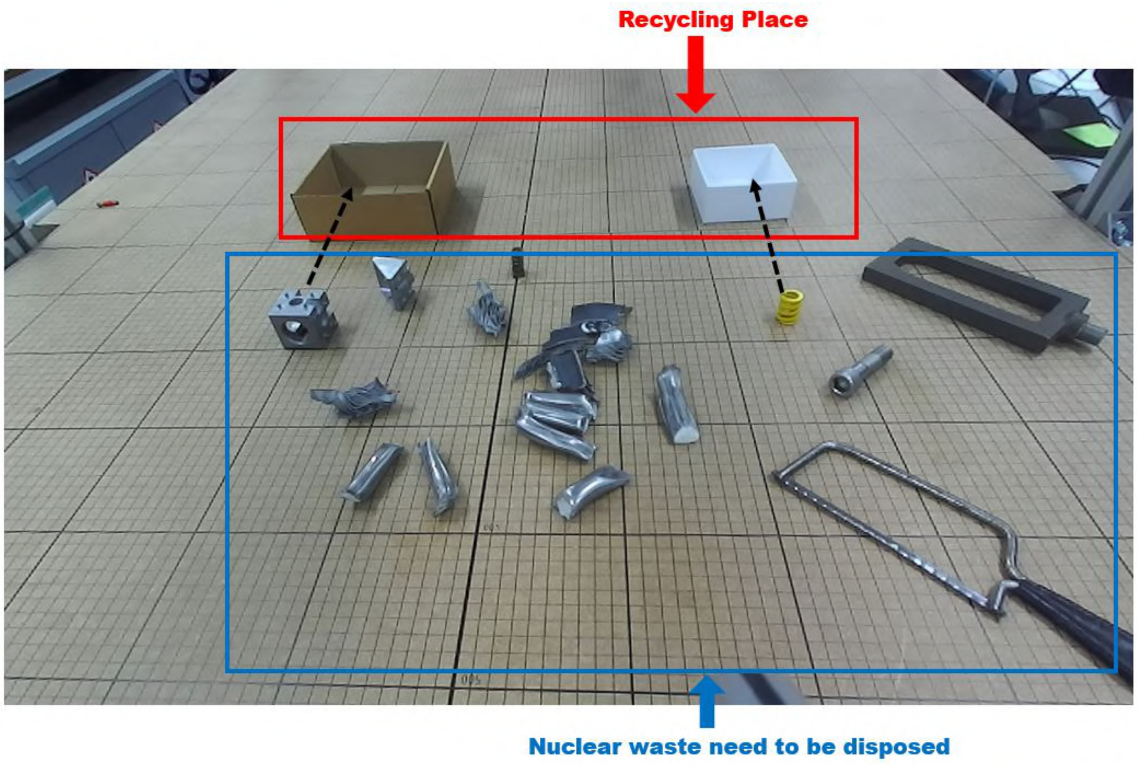


Figure B.7: Simulated nuclear waste hand

The other mode is to utilize an intelligent vision system to finish the task.

For the manual mode, we use the UR5 robot controller panel shown in Figure B.8 to operate the robot to complete the task requested. Arrows in the red and yellow boxes are used to control the pose of the end-effector, and the area with the green box can control the movement for each joint. Note that the robot and gripper systems are driven independent software systems, so we have to drive robot movement in the UR5e controller panel, while controlling the gripper via the computer. This cumbersome operation will imperceptibly increase the workload. In the experiment we conducted, the time to complete the task was 5 minutes and 13 seconds. Apart from this, during the experiment, the pick up for the yellow spring was unsuccessful at the first attempt. Furthermore, the protection procedure was triggered several times due to exceeding the joint limitation. Overall, the fully manual control is not friendly for general users. This mode requires the operator to be very familiar with the spatial movement of the robot and also to have good spatial perception ability. Therefore, this operational mode is more suitable for trained operators as it is prone to inducing significant operator fatigue.

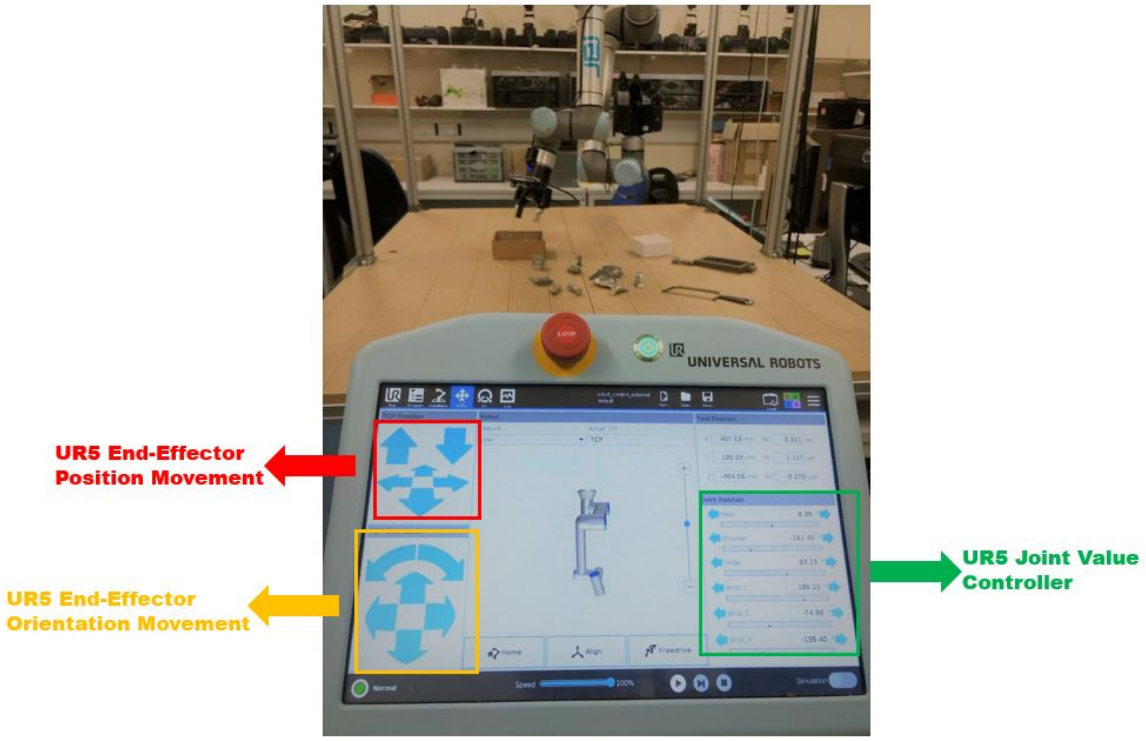

UR5 End-Effector Position Movement
UR5 End-Effector Orientation Movement
UR5 Joint Value Controller
UNIVERSAL ROBOTS

Figure B.8: UR5 robot controller panel

In contrast, the proposed system is more efficient and easy to use. The total time used by the proposed system is 3 minutes and 4 seconds to complete the task, which saves 2 minutes and 9 seconds compared with manual manipulation. In addition, a mouse-and-keyboard robot manipulation system is convenient to most potential operators. Most importantly, the automatic grasping process can free up the operator's hands unlike uninterruptible manual teleoperation. The video of the above two experiments is demonstrated at https://youtu.be/PRu4n-Gs64E

B.4 Summary

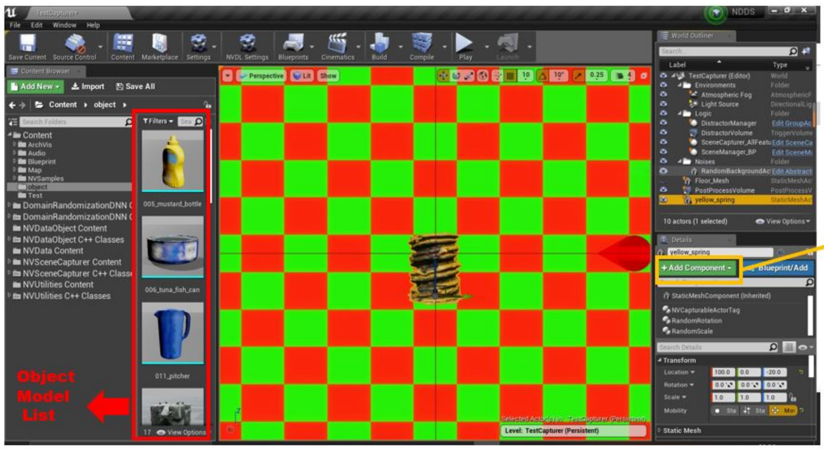
In this chapter we presented a novel robotic vision system that can aid human operators to finish some essential tasks in remote teleoperation. The GUI integrates the various software drivers, environment scanning component, object detection system and robotic manipulation system together to reduce an operator's workload.

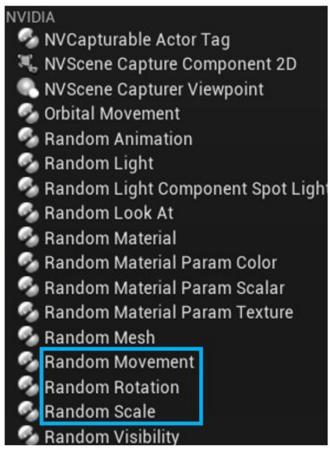
The experimental results show that the proposed teleoperation system is efficient and convenient when compared with manual manipulation via a controller panel. The environment scanning system can provide a sufficiently precise 3D workbench model that allows the operator to perceive the spatial position of the bench from the monitor. Moreover, the proposed system allows the operator to select an object quickly and precisely by automated handling, which improves the efficiency of the operator significantly.

Appendix C

Synthetic Data Generation

This section elaborates the processing of 6D pose synthetic data generation. The core tool is NVIDIA Deep learning Dataset Synthesizer (NDDS) that developed in Unreal Engine 4 environment.

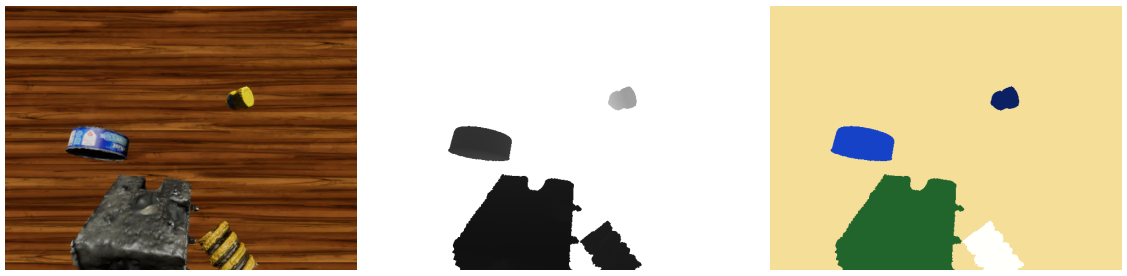



NVIDIA
NVCapturable Actor Tag
NVScene Capture Component 2D
NVScene Capturer Viewpoint
Orbital Movement
Random Animation
Random Light
Random Light Component Spot Light
Random Look At
Random Material
Random Material Param Color
Random Material Param Scalar
Random Material Param Texture
Random Mesh
Random Movement
Random Rotation
Random Scale
Random Visibility

The above figure shows the basic GUI of NDDS. The red box area is the object browser that stores the target object model and texture. Using mouse drag the target model into the middle window. Then in the top right *World Outliner* panel, select the target object name and click *Add Component* button to pop out a menu shown in the right image. In the pop-out menu, there are four component needed to add for 6D pose data generation, which are *NVCapturable Actor Tag*, *NVCapturable Actor Tag*, *Random Movement*, *Random Rotation* and *Random Scale*. The *NVCapturable Actor*

Tag component is used to distinct target object with the background and obstacles. If the scene contains multi instances, we need add *NVCapturable Actor Tag* for each targets. The *Random Movement* can control the position of target objects in the image. We can select the movement range in x, y, and z axis. In our setting, the y range is from $-\frac{image\_wight}{2}$ to $\frac{image\_wight}{2}$, and the z range is from $-\frac{image\_height}{2}$ to $\frac{image\_height}{2}$. This setting can keep objects within the camera's field of view. For x value, we disabled the movement in the x axis because each objects have different dimensions. Alternatively, we adopt *Random Scale* to determine the distance between objects' depth. The range of this parameter is from 0.1 to 10. *Random Rotation* is used to change objects' rotation, and it provides 360 degree rotation at x, y, and z axis.

For the output, NDDS supports RGB image, depth image and segmentation image shown in the figure below. In thesis, we set the size of out put image is 640×480.What's more, NDDS also provides different components for creating highly randomized images, which includes textures, obstacle, lighting, visibility, etc. as top figure listed.


(a) RGB image (b) Depth (c) Instance Segmentation

Apart from image data, NDDS also generates a JASON file that contains camera matrix, camera position, object pose, bounding box, key point etc. data that may be used in training. Overall, using NDDS to generate synthetic data can greatly simplify the process of 6D data annotation which allows researchers to easily create randomized and photorealistic scenes for training deep neural networks.

Bibliography

[1] Seymour A Papert. The summer vision project. 1966.

[2] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[3] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the International Conference on Computer Vision*, pages 858–865. IEEE, 2011.

[4] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[5] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

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

[7] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057. IEEE, 2014.

[8] 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.

[9] 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.

[10] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the International Conference on Computer Vision*, pages 143–152, 2017.

[11] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015.

[12] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision*, pages 1625–1632, 2013.

[13] 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 International Conference on Computer Vision*, pages 1521–1529, 2017.

[14] 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.

[15] 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.

[16] Paul J Besl and Ramesh C Jain. Three-dimensional object recognition. *ACM Computing Surveys (CSUR)*, 17(1):75–145, 1985.

[17] Bir Bhanu and Chih-Cheng Ho. Cad-based 3d object representation for robot vision. *IEEE Annals of the History of Computing*, 20(08):19–35, 1987.

[18] Federico Tombari and Luigi Di Stefano. Object recognition in 3d scenes with occlusions and clutter by hough voting. In *Proceedings of the 2010 Fourth Pacific-Rim Symposium on Image and Video Technology*, pages 349–355. IEEE, 2010.

[19] Anders Glent Buch, Dirk Kraft, Joni-Kristian Kamarainen, Henrik Gordon Petersen, and Norbert Krüger. Pose estimation using local structure-specific shape and appearance context. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2080–2087. IEEE, 2013.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[23] 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.

[24] 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.

[25] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[27] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020.

[28] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 108–125. Springer, 2020.

[29] T To, J Tremblay, D McKay, Y Yamaguchi, K Leung, A Balanon, J Cheng, and S Birchfield. Ndds: Nvidia deep learning dataset synthesizer, 2018.

[30] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019.

[31] Tomas Hodan, Rigas Kouskouridas, Tae-Kyun Kim, Federico Tombari, Kostas Bekris, Bertram Drost, Thibault Groueix, Krzysztof Walas, Vincent Lepetit, Ales Leonardis, et al. A summary of the 4th international workshop on recovering 6d object pose. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[32] Alexandre Lopes, Roberto Souza, and Helio Pedrini. A survey on rgb-d datasets. *arXiv preprint arXiv:2201.05761*, 2022.

[33] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9785–9795, 2019.

[34] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370. IEEE, 2006.

[35] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[36] Christopher Sweeney, Tobias Hollerer, and Matthew Turk. Theia: A fast and scalable structure-from-motion library. In *Proceedings of the International Conference on Multimedia*, pages 693–696, 2015.

[37] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[38] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[39] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011.

[40] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7): 1281–1298, 2011.

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

[42] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision*, 81 (2):155, 2009.

[43] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[44] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3057–3064. IEEE, 2011.

[45] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[46] Changchang Wu et al. Visualsfm: A visual structure from motion system, 2011. *URL http://www. cs. washington. edu/homes/ccwu/vsfm*, 14:2, 2011.

[47] Riccardo Gherardi, Michela Farenzena, and Andrea Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1594–1600. IEEE, 2010.

[48] Hainan Cui, Xiang Gao, Shuhan Shen, and Zhanyi Hu. Hsfm: Hybrid structure-from-motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1212–1221, 2017.

[49] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, pages 3001–3008. IEEE, 2011.

[50] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the International Conference on Computer Vision*, pages 3248–3255, 2013.

[51] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[52] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86. IEEE, 2009.

[53] Tristan Laidlow, Jan Czarnowski, and Stefan Leutenegger. Deepfusion: real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4068–4074. IEEE, 2019.

[54] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5(2):721–728, 2020.

[55] 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 *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[56] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. 1996.

[57] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[58] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John J Leonard, and John McDonald. Kintinuous: Spatially extended kinectfusion. 2012.

[59] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2013.

[60] 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.

[61] Simone Fantoni, Umberto Castellani, and Andrea Fusiello. Accurate and automatic alignment of range surfaces. In *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 73–80. IEEE, 2012.

[62] Ajmal S Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1584–1601, 2006.

[63] Gregory T Flitton, Toby P Breckon, and Najla Megherbi Bouallagu. Object recognition using 3d sift in complex ct volumes. In *Proceedings of the British Machine Vision Virtual Conference*, volume 1, pages 1–12, 2010.

[64] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Proceedings of the International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. IEEE, 2009.

[65] Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27 (11):963–976, 2011.

[66] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the European Conference on Computer Vision*, pages 356–369. Springer, 2010.

[67] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proceedings of the International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[68] 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 *Proceedings of the International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592. IEEE, 2011.

[69] Marius Muja and David Lowe. Flann-fast library for approximate nearest neighbors user manual. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, 5, 2009.

[70] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques*, pages 111–120, 2006.

[71] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *Proceedings of the European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[72] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

[73] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Proceedings of the International Conference on 3D Vision*, pages 239–248. IEEE, 2016.

[74] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision*, pages 628–644. Springer, 2016.

[75] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[76] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2690–2698, 2019.

[77] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3244–3253, 2019.

[78] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *arXiv preprint arXiv:1708.05375*, 2017.

[79] Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and CL Philip Chen. 3d2seqviews: Aggregating sequential views for 3d global feature learning by cnn with hierarchical attention aggregation. *IEEE Transactions on Image Processing*, 28(8):3986–3999, 2019.

[80] Meng Wang, Lingjing Wang, and Yi Fang. 3densinet: A robust neural network architecture towards 3d volumetric object prediction from 2d image. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 961–969, 2017.

[81] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017.

[82] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 607–623, 2018.

[83] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8966, 2019.

[84] Sheng Ao, Qingyong Hu, Bo Yang, Andrew Markham, and Yulan Guo. Spinnet: Learning a general surface descriptor for 3d point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11753–11762, 2021.

[85] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 153–161, 2017.

[86] 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.

[87] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2514–2523, 2020.

[88] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7193–7203, 2020.

[89] Caner Sahin and Tae-Kyun Kim. Recovering 6d object pose: a review and multimodal analysis. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[90] Emal Sadran, Kai M Wurm, and Darius Burschka. Sparse keypoint models for 6d object pose estimation. In *Proceedings of the European Conference on Mobile Robots*, pages 307–312. IEEE, 2013.

[91] Thilo Grundmann, Robert Eidenberger, Martin Schneider, Michael Fiegert, and Georg v Wichert. Robust high precision 6d pose determination in complex environments for robotic manipulation. In *Proceedings of the International Conference of Robotics and Automation*, pages 1–6, 2010.

[92] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6702–6710, 2017.

[93] 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.

[94] 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.

[95] Yinlin Hu, Pascal Fua, Wei Wang, and Mathieu Salzmann. Single-stage 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2930–2939, 2020.

[96] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image. *arXiv preprint arXiv:1802.10367*, 2018.

[97] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3385–3394, 2019.

[98] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16611–16621, 2021.

[99] Tomáš Hodaň, Xenophon Zabulis, Manolis Lourakis, Štěpán Obdržálek, and Jiří Matas. Detection and fine 3d pose estimation of texture-less objects in rgb-d images. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4421–4428. IEEE, 2015.

[100] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Proceedings of the European Conference on Computer Vision*, pages 536–551. Springer, 2014.

[101] Huixu Dong, Dilip K Prasad, and I-Ming Chen. Object pose estimation via pruned hough forest with combined split schemes for robotic grasp. *IEEE Transactions on Automation Science and Engineering*, 2020.

[102] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11824–11833, 2020.

[103] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[104] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

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

[106] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Frank Perbet, and Björn Stenger. Demisting the hough transform for 3d shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341, 2014.

[107] Mathieu Gonzalez, Amine Kacete, Albert Murienne, and Eric Marchand. Yoloff: You only learn offsets for robust 6dof object pose estimation. *arXiv preprint arXiv:2002.00911*, 2020.

[108] 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.

[109] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4404–4413, 2020.

[110] Catherine Capellen, Max Schwarz, and Sven Behnke. Convposecnn: Dense convolutional 6d object pose estimation. *arXiv preprint arXiv:1912.07333*, 2019.

[111] Haoruo Zhang and Qixin Cao. Texture-less object detection and 6d pose estimation in rgb-d images. *Robotics and Autonomous Systems*, 95:64–79, 2017.

[112] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *Proceedings of the European Conference on Computer Vision*, pages 462–477. Springer, 2014.

[113] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

[114] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4631–4640, 2017.

[115] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[116] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21, 2019.

[117] Ge Gao, Mikko Lauri, Xiaolin Hu, Jianwei Zhang, and Simone Frintrop. Cloudaae: Learning 6d object pose regression with on-line data synthesis on point clouds. *arXiv preprint arXiv:2103.01977*, 2021.

[118] Weitong Hua, Jiaxin Guo, Yue Wang, and Rong Xiong. 3d point-to-keypoint voting network for 6d pose estimation. In *Proceedings of the 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 536–541. IEEE, 2020.

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

[120] Jaesik Chang, Minju Kim, Seongmin Kang, Heungwoo Han, Sunpyo Hong, Kyunghun Jang, and Sungchul Kang. Ghostpose*: Multi-view pose estimation of transparent objects for robot hand grasping. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5749–5755. IEEE.

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

[122] Pengyuan Wang, HyunJun Jung, Yitong Li, Siyuan Shen, Rahul Parthasarathy Srikanth, Lorenzo Garattoni, Sven Meier, Nassir Navab, and Benjamin Busam. Phocal: A multi-modal dataset for category-level object pose estimation with photometrically challenging objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21222–21231, 2022.

[123] Song Ge and Guoliang Fan. Articulated non-rigid point set registration for human pose estimation from 3d sensors. *Sensors*, 15(7):15218–15245, 2015.

[124] Félix Nadon, Angel J Valencia, and Pierre Payeur. Multi-modal sensing and robotic manipulation of non-rigid objects: A survey. *Robotics*, 7(4):74, 2018.

[125] Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, Niki Trigoni, and Andrew Markham. 3d-physnet: Learning the intuitive physics of non-rigid object deformations. *arXiv preprint arXiv:1805.00328*, 2018.

[126] Félix Nadon and Pierre Payeur. Automatic selection of grasping points for shape control of non-rigid objects. In *Proceedings of the 2019 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 1–7. IEEE, 2019.

[127] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[128] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. ipose: instance-aware 6d pose estimation of partly occluded objects. In *Proceedings of the Asian Conference on Computer Vision*, pages 477–492. Springer, 2018.

[129] Amey Kasar. Benchmarking and comparing popular visual slam algorithms. *arXiv preprint arXiv:1811.09895*, 2018.

[130] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018.

[131] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2015.

[132] Martin Brossard, Silvere Bonnabel, and Axel Barrau. A new approach to 3d icp covariance estimation. *IEEE Robotics and Automation Letters*, 5(2):744–751, 2020.

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

[134] Michael Korn, Martin Holzkothen, and Josef Pauli. Color supported generalized-icp. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 3, pages 592–599. IEEE, 2014.

[135] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.

[136] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[137] Yao Jianchao and Chia Tien Chern. Comparison of newton-gauss with levenberg-marquardt algorithm for space resection. In *Proceedings of the 22nd Asian Conference on Remote sensing*, pages 5–9, 2001.

[138] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[139] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. IEEE, 2008.

[140] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):698–700, 1987.

[141] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[142] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016.

[143] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6d object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 462–471, 2017.

[144] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[145] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2011.

[146] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017.

[147] Tomas Hodan, Daniel Barath, and Jiri Matas. Epos: estimating 6d pose of objects with symmetries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2020.

[148] 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.

[149] Alex M Andrew. Multiple view geometry in computer vision. *Kybernetes*, 2001.

[150] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

[151] Jiale Chen, Lijun Zhang, Yi Liu, and Chi Xu. Survey on 6d pose estimation of rigid object. In *Proceedings of the 2020 39th Chinese Control Conference (CCC)*, pages 7440–7445. IEEE, 2020.

[152] T Do, Trung Pham, Ming Cai, and Ian Reid. Real-time monocular object instance 6d pose estimation. 2019.

[153] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[154] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830, 2011.

[155] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[156] Wanqing Zhao, Shaobo Zhang, Ziyu Guan, Hangzai Luo, Lei Tang, Jinye Peng, and Jianping Fan. 6d object pose estimation via viewpoint relation reasoning. *Neurocomputing*, 2020.

[157] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3d object detection and pose estimation. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition Workshops, pages 2038–2041, 2018.

[158] 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 *Proceedings of the European Conference on Computer Vision*, pages 205–220. Springer, 2016.

[159] Meng Tian, Liang Pan, Marcelo H Ang, and Gim Hee Lee. Robust 6d object pose estimation by learning rgb-d features. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6224. IEEE, 2020.

[160] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[161] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[162] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1941–1950, 2019.

[163] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[164] Sandor M Veres. Natural language programming of agents and robotic devices. 2008.

[165] Sandor M Veres. Theoretical foundations of natural language programming and publishing for intelligent agents and robots. *Proc. TAROS*, pages 678–687, 2010.

[166] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[167] Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *Proceedings of the 2019 Third IEEE International Conference on Robotic Computing*, pages 590–595. IEEE, 2019.

[168] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5): 421–436, 2018.

[169] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. In *Proceedings of the Conference on Robot Learning*, pages 281–290. PMLR, 2017.

[170] Lei Tai, Jingwei Zhang, Ming Liu, Joschka Boedecker, and Wolfram Burgard. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. *arXiv preprint arXiv:1612.07139*, 2016.

[171] Sebastian Thrun and Lorien Pratt. *Learning to learn.* Springer Science & Business Media, 2012.

[172] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 1, page 2, 2000.

[173] Hans Bruun Nielsen et al. *Damping parameter in Marquardt's method.* IMM, 1999.

[174] Thomas B Sheridan. Human–robot interaction: status and challenges. *Human factors*, 58(4):525–532, 2016.

[175] Günter Niemeyer, Carsten Preusche, Stefano Stramigioli, and Dongjun Lee. Telerobotics. In *Springer Handbook of Robotics*, pages 1085–1108. Springer, 2016.

[176] Naresh Marturi, Alireza Rastegarpanah, Chie Takahashi, Ma xime Adjigble, Rustam Stolkin, Sebastian Zurek, Marek Kopicki, Mohammed Talha, Jeffrey A Kuo, and Yasemin Bekiroglu. Towards advanced robotic manipulation for nuclear decommissioning: A pilot study on tele-operation and autonomy. In *Proceedings of the 2016 International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, pages 1–8. IEEE, 2016.

[177] Victor A Prisacariu and Ian D Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. *International Journal of Computer Vision*, 98(3):335–354, 2012.

[178] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[179] Matthew Cousins, Chenguang Yang, Junshen Chen, Wei He, and Zhaojie Ju. Development of a mixed reality based interface for human robot interaciotn. In *Proceedings of the 2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1, pages 27–34. IEEE, 2017.

[180] Jeffrey I Lipton, Aidan J Fay, and Daniela Rus. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*, 3(1):179–186, 2017.

[181] Lorenzo Peppoloni, Filippo Brizzi, Carlo Alberto Avizzano, and Emanuele Ruffaldi. Immersive ros-integrated framework for robot teleoperation. In *Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI)*, pages 177–178. IEEE, 2015.

[182] Dejing Ni, AYC Nee, SK Ong, Huijun Li, Chengcheng Zhu, and Aiguo Song. Point cloud augmented virtual reality environment with haptic constraints for teleoperation. *Transactions of the Institute of Measurement and Control*, 40(15): 4091–4104, 2018.

[183] Balazs Vagvolgyi, Wenlong Niu, Zihan Chen, Paul Wilkening, and Peter Kazanzides. Augmented virtuality for model-based teleoperation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3826–3833. IEEE, 2017.