Learning Markerless Robot-Depth Camera Calibration and End-Effector Pose Estimation

by

Buğra Can Sefercik

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

August 5, 2022

Learning Markerless Robot-Depth Camera Calibration and End-Effector Pose Estimation

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Buğra Can Sefercik

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Assist. Prof. Barış Akgün (Advisor)

Prof. Yücel Yemez

Assoc. Prof. Emre Uğur

Date:

To my beloved family

ABSTRACT

Learning Markerless Robot-Depth Camera Calibration and End-Effector Pose Estimation
Buğra Can Sefercik
Master of Science in Computer Science and Engineering
August 5, 2022

Robot arms are being used more and more in unstructured environments. As such, they are relying more on vision sensors compared to traditional factory robots which are placed in highly structured, controlled and caged work-cells. Some applications that rely on vision data include bin-picking, box picking and placing, assembly and part feeding in mixed human-robot work cells, inspection, quality control etc. Vision data is mainly required to localize the objects to be manipulated, perform measurements and detect near-by humans. Vision based robot systems require extrinsic calibration between the robot and camera in order to work properly. This is a time consuming and tedious procedure which can be expensive as well. Fast, flexible and precise robot-camera calibration is essential for not only industrial environments but also academic lab environments where the location of the camera and/or robot needs to be frequently or is accidentally changed.

Extrinsic calibration between a robot arm and camera is a decades old challenge still prevalent to this day. Traditional techniques work by estimating pose of the camera relative to a fiducial marker from multiple points and matching these estimations with the robot's pose. Recent learning based approaches predict extrinsic calibration from images relying heavily on simulation data.

In this thesis, we present a learning based markerless extrinsic calibration system that uses a depth camera. We learn models for end-effector (EE) segmentation, single-frame rotation prediction and keypoint detection, from automatically generated real-world data. Our models are based on MinkUNet and PointNet++ architectures. We use a transformation trick to get EE pose estimates from rotation predictions and a matching algorithm to get EE pose estimates from keypoint predictions. We further utilize the iterative closest point (ICP) algorithm, multiple-frames and outlier detection to increase calibration robustness. Our results on the

test set with previously unseen camera locations give sub-centimeter ($0.74cm$) and less than 0.05 radians ($1.69°$) average calibration errors and $1.00cm$ and $2.74°$ average pose estimation errors. In addition, we released an open source easy to use tool for robot users to handle robot-camera calibration with a few mouse clicks by seamlessly integrating all the models and algorithms discussed in this thesis.

ÖZETÇE

Öğrenim Tabanlı Robot-3B Kamera Harici Kalibrasyonu ve Uç Efektör Pozisyonu Tahmini
Buğra Can Sefercik
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
5 Ağustos 2022

Robot kolları yapılandırılmamış ortamlarda giderek daha fazla kullanılıyor. Bu nedenle, yüksek düzeyde yapılandırılmış, kontrollü ve kafesli iş hücrelerine yerleştirilen geleneksel fabrika robotlarına kıyasla görüntü sensörlerine daha fazla ihtiyaç duyuyorlar. Çöp toplama, kutu taşıma ve yerleştirme, insan-robot ortak çalışma hücrelerinde montaj ve parça besleme, inceleme, kalite kontrol sistemleri görüntü sensörü kullanan uygulamalara örnek olarak gösterilebilir. Görüntü verileri esas olarak manipüle edilecek nesnelerin yerini belirlemek, ölçümler yapmak ve yakındakı insanları tespit etmek için gereklidir. Görüntü tabanlı robot sistemlerinin düzgün çalışabilmesi için, robot ve kamera arasında harici kalibrasyon işlemi yapılması gerekir. Bu, zaman alıcı ve sıkıcı aynı zamanda da pahalı olabilen bir prosedürdür. Hızlı, esnek ve hassas robot-kamera kalibrasyonu, yalnızca endüstriyel ortamlar için değil, aynı zamanda kamera ve/veya robotun konumunun sık sık değiştirilmesi gereken veya yanlışlıkla değiştirildiği akademik laboratuvar ortamları için de gereklidir.

Robot-kamera harici kalibrasyonu, on yıllardır süre gelen ve bugün dahi üzerinde çalışılan bir problem. Geleneksel yöntemler, referans bir nesnenin kameraya göre afin dönüşümünü farklı robot pozisyonlarından hesaplayarak ve bu afin dönüşüm tahminlerini robotun yön ve konumuyla eşleştirerek çalışır. Öğrenme tabanlı güncel yaklaşımların büyük bir kısmı simülasyon ortamında oluşturulan verileri kullanarak harici kalibrasyon tahminini gerçekleştiriyor.

Bu tez çalışmasında; referans nesneye ihtiyaç duymayan, öğrenme tabanlı bir robot-3B kamera harici kalibrasyon sistemi sunuyoruz. Otomatik olarak oluşturulan gerçek dünya verilerilerini kullanarak, uç efektör (EE) segmentasyonu, EE yön tahmini ve kilit nokta tespiti için modeller eğitiyoruz. Öğrenme modellerimiz MinkUNet ve PointNet++ mimarileri kullanılarak oluşturuldu. EE konum hesaplamasını, EE segmentasyon ve EE yön modellerinden elde edilen tahminlerini kullanarak yapıyoruz.

Tespit edilen kilit noktaları referans kilit noktalarıyla eşleyerek, EE yön ve konum tahminini ikinci bir yöntemle de gerçekleştirmiş oluyoruz. Kalibrasyon kalitesini artırmak için yinelemeli yakın nokta (ICP) algoritmasından, birden fazla 3B fotoğraf verisinden ve aykırı değer analizinden faydalanıyoruz. Daha önce kullanılmamış test verileriyle yaptığımız ölçümlerde; 1 santimetreden daha az ($0.74cm$) konum ve 0.05 radyandan daha az ($1.69°$) yön ortalama hata paylarıyla kalibrasyon işlemini gerçekleştirdik. Tek kareden yaptığımız EE yön ve konum hesaplamarında, $1.00cm$ konum and $2.74°$ yön ortalama hata paylarına ulaştık. Ayrıca, bu tezde anlatılan tüm model ve algoritmaları entegre ederek; açık kaynaklı, kullanıcı dostu robot-3B kamera harici kalibrasyon yazılımını geliştirdik.

ACKNOWLEDGMENTS

It was a great privilege to have Assist. Prof. Barış Akgün as my thesis advisor. Thank you for being incredibly patient with me and believing in me even when I wasn't. This work wouldn't have been possible without his help and guidance. The insights that I gained while working with him will always help me through my career.

I want to thank Çağlar Kılcıoğlu, Uğur Halatoğlu, Serkan Öncül and Mustafa Ozan Çapa for encouraging me to pursue a master's degree and helping me collect myself when I got discouraged. They mean more to me than mere colleagues.

I am grateful for Farzin Negahbani and Onur Berk Töre's help through my entire master education. They never hesitated to help me when I came short during the experiments and classes. Thank you for being such good, helpful friends in this journey.

Last but not least, I want to thank my family. Without their endless support and encouragement, none of my accomplishments would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DoF	Degrees of Freedom
EE	End Effector
FOV	Field of View
ICP	Iterative Closest Point
KPM	Keypoint Matching
PnP	Perspective-n-Point
POV	Point of View
RPT	Rotation Prediction and Transform

Chapter 1

INTRODUCTION

Robot arms have been deployed as part of automation systems in factory floors since 1970s. Robotic automation systems are precise, repeatable, fast and they reduce human workload both literally and figuratively. These systems are highly engineered, customized and expensive to setup. They are mainly housed in caged work-cells since they are not safe during operation. As such, these systems are mostly feasible for mass production.

There are practical, economical and intellectual drives for broader applicability of robot arms which brought about the advent of "collaborative" robots or cobots. Cobots are designed to be safer to negate the need for cages, more cost effective (both in terms of unit and integration costs) and easily re-programmable by sacrificing payload capacity and to an extent precision. Some tasks that these cobots are tasked with include parts manipulation, bin-picking, box picking-and-placing, and assembly. These tasks require the localization of manipulation targets with respect to the robot. A popular approach is to use vision towards this end where a camera estimates the pose of the target with respect to itself which is then transformed to the robot's coordinate system. The last step requires the knowledge of the camera with respect to the robot which is called **extrinsic calibration**.

Keeping the correct calibration is a challenge for robots in dynamic and uncontrolled environments encountered by cage-free robots. Even if the robot and the camera are meant to be static. Multiple factors alter this calibration such as re-positioning the robot and/or the camera for better workspace coverage or for different applications (e.g. research projects), inadvertently moving them for cleaning, people bumping into camera rigs and robots, wear-and-tear and backlash on

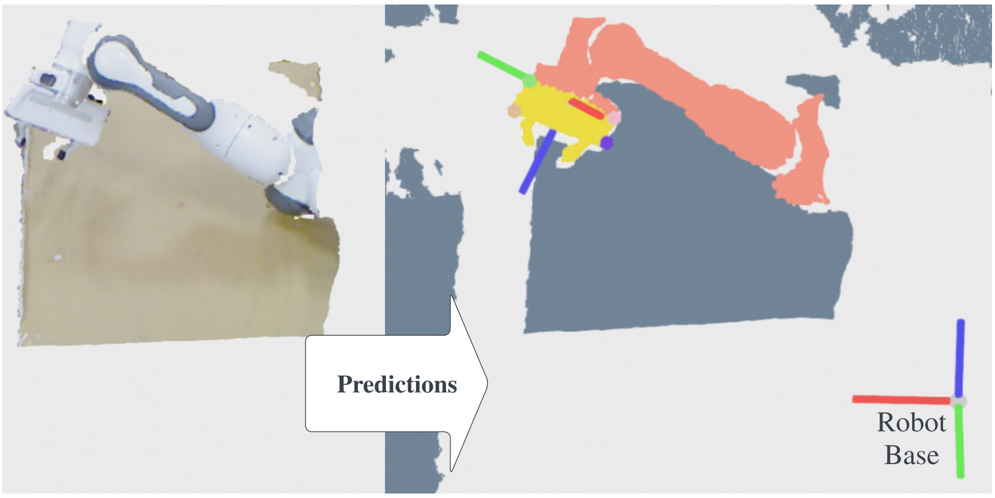


Figure 1.1: Sample point cloud input of the system and predictions for the input.

low-cost fixtures etc. As such calibration is a needed but time-consuming process. At the very least, calibration needs to be double-checked before the real work can begin. This is an all-too-real issue for robotic researchers working with robot arms, which is getting more wide-spread as cage-free robot arms and mobile manipulators become more common in the industry.

Traditional extrinsic calibration approaches rely on fiducial markers. Checkerboard patterns and augmented reality tags are commonly used in both research and factory settings. Adding markers is error-prone due to intrinsic calibration errors, sensor noise, robot-to-marker fixture quality (which introduces errors in transformation between the markers and the robot) etc. Almost all the approaches benefit from post-processing steps to improve the calibration quality such as the iterative closest point (ICP) algorithm, if a reference model is available. High precision systems employ precision machined fixtures and active markers (e.g. leds, infrared reflective materials along with infrared light-sources) which are costly. Thus, using markers is not an ideal solution for most applications.

In this work, we develop a system that can handle the extrinsic calibration between a depth camera and a robot arm without additional hardware, markers or simulation. We utilize recent developments in deep learning models for point clouds in combination with the ICP algorithm. Our system collects its own data, negating the need for manual labelling. Two side benefits of our system is that (1) it provides high quality semantic segmentation information about the background, end-effector and the rest of the robot, and (2) estimates the end-effector (EE) pose with high accuracy.

1.1 Thesis Statement

It is possible to build a fast and flexible learning based markerless extrinsic calibration system using a noisy consumer-grade depth camera, with minimum human intervention, and without any additional equipment that can achieve sub-centimeter level position and sub-deciradian level rotation precision, using only real data.

1.2 Contributions

The main contributions of our system are;

(1) high performance (average test errors of $0.74cm$ and $1.69°$) markerless extrinsic calibration,

(2) high performance (average test errors of $1.00cm$ and $2.74°$) EE pose estimation with respect to camera,

(3) automatic real-life data collection methodology,

(4) an easy-to-use, open-source robot-depth camera calibration tool.

1.3 Outline

Remaining chapters of this thesis are organized as follows:

- Ch. 2: investigates the conventional and learning-based robot-camera calibration literature.

- Ch. 3: gives background information on the available technical methods utilized in this work. Experienced readers are encouraged to skip this chapter.

- Ch. 4: explains our setup and our automatic data collection methodology.
- Ch. 5: explains our novel single frame end-effector pose estimation method.
- Ch. 6: explains how single frame end-effector pose estimations can be used for precise robot-camera calibration.
- Ch. 7: presents our experimental setup and discusses results of our experiments.
- Ch. 8: summarizes the thesis and highlights the possible future directions.
- Appx. A: provides neural network structures, training details and describes the outlier rejection approach.

Chapter 2

LITERATURE REVIEW

Calculating the transformation between a robot's base and a camera, usually called extrinsic calibration, is a well-established problem. Foundations of prevailing robot-camera calibrations techniques were laid in the late 1980's. Essentially, the calibration task is to calculate the pose of the robot's EE or robot's base with respect to the camera that is either mounted on the EE (eye-in-hand) or placed nearby the robot arm (eye-to-hand) as shown in Fig. 4.1a. The ultimate goal of the robot-camera calibration is to get the pose of a target (e.g. an object), which is estimated by using the camera data, in the robot's base frame to facilitate robot tasks such as manipulation, avoidance etc.

In this chapter, we comprehensively investigate how conventional and learning based methods tackle the robot-camera calibration problem as well as several related approaches.

2.1 Conventional Approaches

Conventional techniques use fiducial markers to estimate camera to robot transformations. Most of the popular approaches attach a fiducial marker on the robot and/or place a marker [Garrido-Jurado et al., 2014, Fiala, 2005, Olson, 2011] in the vicinity of the robot's working envelope. [Tsai and Lenz, 1988] formulates 3D robot-camera calibration as we know today building upon [Shiu and Ahmad, 1987]'s work. It establishes calibration methods for both eye-to-hand and eye-in-hand setups based on the use of a calibration object. In [Tsai and Lenz, 1988], a 2D camera attached to the robot wrist in eye-in-hand setup and two 2D cameras placed in the proximity of robot's working envelope in 3D eye-to-hand setup. Calibration is done using kinematic information and collecting multiple images from different robot joint configurations providing that the static calibration object is visible to the camera(s). [Zhan and Wang, 2012] develops an offline calibration system and online calibration system for an industrial eye-in-hand robot drilling system adopting the principles and techniques devised by [Shiu and Ahmad, 1987, Tsai and Lenz, 1988] 30+ years ago. [Liu et al., 2020] presents a 3D surface scanner system that uses a fixed 2D dotted calibration surface following the same method to periodically optimize its scanner and calibration model parameters during scanning. [Kroeger et al., 2019] describes a four-camera eye-to-hand system that uses fiducial markers for not only extrinsic camera calibration but also automated intrinsic stereo calibration relying on the similar methods. Intrinsic calibration is handled by moving the robot to various joint configurations having placed a fiducial marker between its grippers and extrinsic camera calibration is done by sticking multiple marker tags on robot base.

[Pauwels and Kragic, 2016]'s work is based on the use of everyday objects such as canned/boxed goods instead of conventional markers provided that 3D models of the objects are available in advance. The object is placed inside the gripper and robot base calibration parameters are periodically calculated and optimized using object registration and forward kinematics information while the robot arm is moving.

Major disadvantages of using such markers include wear-and-tear, distance dependent camera noise [Wasenmüller and Stricker, 2016], fixture backlash and build quality etc. For instance, [Pauwels and Kragic, 2016] uses a Pringless can as a fiducial marker whose 3D model was published by [Calli et al., 2015]; however, chips producer changed the coating of the can in 2021 which made use of the available 3D model obsolete for calibration procedure. There are also single frame methods that use fiducial markers to estimate the EE pose [Peng et al., 2020, Pauwels and Kragic, 2016] in real-time.

Marker based methods start to fail when the reference objects/markers are not as visible enough as anticipated or somehow the marker model changes. Some of the marker based solutions also suffer from marker position uncertainty (e.g. when markers are grasped by the end effector) There are expensive and specialized, often high-precision, expensive and active (e.g. IR markers), systems used in the industry which are not feasible for all robot applications. In this work, we aim to develop a markerless approach to handle extrinsic calibration.

2.2 Learning Based Approaches

The advent of deep learning brought breakthroughs in virtually every computer vision task that has been around for decades. Naturally, these developments apply to both robot-camera calibration and EE pose estimation. Other computer vision tasks that are closely related to the methods discussed in this thesis include semantic segmentation, 6-DoF object pose estimation and keypoint (landmark) detection.

[Lee et al., 2020, Labbé et al., 2021] have produced successful results on robot pose estimation from a single frame for eye-to-hand setups. [Lee et al., 2020] learns to estimate hand-picked keypoints (joint connection points) of entire robot arm from an RGB image trained with the dataset generated in various simulation environments. They combine forward kinematics information with the predictions to estimate the robot pose in 3D space using Perspective-n-Point algorithm (PnP). [Labbé et al., 2021] approaches the same problem from a different point of view though using the same RGB data with or without forward kinematics information. The method takes an RGB image and generates initial 6D pose and joint state estimates. Then, they iteratively render a CAD model using the joint state predictions and apply CAD to image matching to optimize 6D pose and joint state estimates. They also directly estimate 6D robot pose provided that the joint states are already available from forward kinematics. [Lambrecht, 2019] adopts a CNN based few-shot method for robot pose estimation. Similar to [Lee et al., 2020], they learn to predict manually selected robot keypoints from RGB images, then estimate robot pose using keypoint predictions and forward kinematics. Our system diverges from these methods in the following areas: (1) they require majority of the robot arm to be visible while our only requirement is to see the EE in the frame and part of its adjacent robot link, (2) our method uses depth data whereas others rely on 2D to 3D projection using the PnP algorithm to compute affine transformations, hence, we can readily apply the ICP algorithm to refine our pose estimates and (3) we do not require any synthetic/simulation data and use only real-world data. It can be argued that we are solving a simpler problem (only EE and depth data) in exchange for higher calibration quality.

Partially similar to our system, [Valassakis et al., 2021] introduces a learning based solution to eye-in-hand pose estimation problem using single RGB frame. It learns to directly regress camera to EE affine transformation parameters using CNNs via simulation data. Compared to our method, this approach solely relies synthetic data and 2D images, and works in a different robot-camera setup (eye-in-hand).

[Hwang et al., 2020, Peng et al., 2020] utilize the conventional methods and learning based methods in combination. They both attach fiducial markers (set of 3 spheres with different colors) to the EE of a surgical robot and estimate initial affine transformation using conventional techniques that we discussed in Sect. 2.1. Then, [Hwang et al., 2020] uses an RNN model to reduce tracking error due conventional method's estimation while [Peng et al., 2020] feeds an MLP with the initial estimation and kinematics information to enhance instant predictions. In contrast to these methods, we use only learning based methods for initial estimations and don't rely on fiducial markers.

2.3 Related Tasks

2.3.1 6-DoF Object Pose Estimation

6-D object pose estimation is a fundamental task in robotics as knowing the pose of an object with respect to a camera is a mandatory step for manipulation. We apply similar ideas to estimate the pose of the end-effector with respect to the camera to be later used in extrinsic calibration. The field of 6D pose estimation is broad and thus we only mention the methods we use.

We perform EE rotation prediction using the techniques presented in [Xiang et al., 2018, Li et al., 2018b]. They both learn object rotation prediction by using a custom version of least squares fitting target as a loss function. In order to predict the object translation, [Xiang et al., 2018] reconstructs the predictions for two RGB frames using intrinsic calibration information and [Li et al., 2018b] trains an RNN model to predict translation by iteratively predicting and optimizing registration between the reference object and the target object. Although we train our rotation model with the same loss function, we employ a completely different approach for translation prediction.

2.3.2 Semantic Segmentation

Semantic segmentation is an essential part of most computer vision applications ranging from background replacement in online meetings to brain tumor identification and localization. Objective of semantic segmentation is to convert raw image representation to a simpler, category-based representations by labeling each image pixel with a certain category. There are numerous publicly-available off-the-shelf 2D semantic segmentation architectures from which selection can be done based on one's needs such as speed, resolution and accuracy [He et al., 2017, Ronneberger et al., 2015, Sandler et al., 2018]. Even though MLP based [Qi et al., 2017a, Qi et al., 2017b], and CNN based [Li et al., 2018a] architectures were published for point cloud processing, point cloud based semantic segmentation has ramped up with the introduction of the sparse convolutional networks [Liu et al., 2015, Jiang et al., 2020, Cheng et al., 2021, Vu et al., 2022].

In this work, we utilize a sparse convolutional backbone [Choy et al., 2019] to segment robot EE, robot arm and background from point cloud inputs.

2.3.3 3D Object Keypoint Detection

Keypoint (landmark) detection is a well-studied task in RGB space, specially for human bodies and faces [Kazemi and Sullivan, 2014, Huber et al., 2016, Toshev and Szegedy, 2014]. Keypoints are used for affine transformation calculation between a reference object and a target object. Although there have been decent solutions in 2D space for many years, 3D methods that are not dependent on PnP transformation from 2D images have emerged only recently. [You et al., 2020] published a 3D object dataset having manually selected keypoint labels with benchmark results for commonly used 3D neural backbones. [Jakab et al., 2021] introduces an unsupervised keypoint discovery algorithm for object point clouds.

In this work, we train a neural network to estimate manually picked keypoints for the EE and use the keypoints to compute affine transformation entirely in 3D space based on the ideas from these methods.

2.4 Summary

In this chapter, we introduced some of the prevalent conventional and learning based techniques in robot-camera calibration literature. Conventional methods rely on markers and solving the homogeneous transformation equation. Learning based methods mainly utilize simulation data. Some of them directly estimate the calibration whereas some solve the equation. We also briefly presented pose estimation, learning based semantic segmentation and keypoint detection methods, and discussed how they serve our purposes.

Chapter 3

BACKGROUND

3.1 Robot Coordinate Systems

A robot system generally consists of a mechanical robot arm with several links and axes, an end-effector attached to the edge of the arm such as a gripper, a driller or a hand model along with a computer controller responsible for moving the robot arm and providing the communication between the robot and workstation computers running ROS. Most robot arms have 6 or 7 DoF, in other words have 6 or 7 single-axis rotational joints to let them move the EE to various locations with different translation and rotation configurations. Franka Emika Panda robot arm has 7 DoF as show in Fig. 4.1b.

In robot systems, not only robot joint states but also other objects and camera are assigned with individual 3D coordinate frames, or poses, containing both translation and rotation information with respect to a reference coordinate frame as shown in Fig. 3.1. For example, the robot base pose is given with respect to world frame whereas object poses are given with respect to user frame in the Fig. 3.1. Since these coordinate frames hold both translation and rotation data, we need a unified mathematical description of these relative transformations for computational consistency. Let p_a and p_b denote geometric coordinate vectors, i.e. x, y, z, for frame $\{a\}$ and frame $\{b\}$, respectively, in the same coordinate system. Then, p_a can computed as in Eq. 3.1:

$$p_a = R^a_b \times p_b + t^a_b \tag{3.1}$$

where R^a_b denote rotation matrix of $\{a\}$ with respect to $\{b\}$ and t^a_b denote translation vector of $\{a\}$ with respect to $\{b\}$.

Eq. 3.1 can be written as an affine transformation operation where homogeneous transformation matrix T^a_b denotes the rotation and transformation of frame $\{a\}$

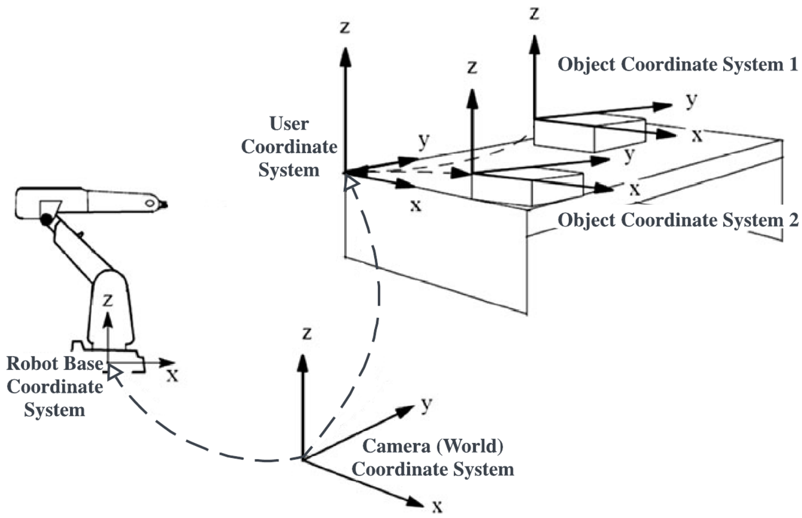


Figure 3.1: ABB Robot co ordinate system visualization [Deng et al., 2012].

whose origin is displaced from frame $\{b\}$.

$$T_b^a = \begin{bmatrix} R_b^a & t_b^a \\ 0^T & 1 \end{bmatrix} \tag{3.2}$$

Using this notation, it is now easier and more organized to calculate geometrical coordinates of the frame $\{a\}$:

$$\begin{bmatrix} p_a \\ 1 \end{bmatrix} = T_b^a \times \begin{bmatrix} p_b \\ 1 \end{bmatrix} \tag{3.3}$$

3.2 Quaternion Notation

In Sect. 3.1, we denoted 3D orientation information with 3x3 rotation matrices. Two other popular rotation notations in robotics applications are Euler angles and quaternions. The Euler angles are the three angles denoting the rotational deviation from x, y, z axes angles. They are much more intuitive than other notations; however, they are constricted by the gimbal lock phenomenon occurring when two of the three axes are driven into a parallel alignment causing dimensional reduction. This phenomenon prevents orientation angles from getting close to 90°. Constrastingly, quaternion notation does not suffer from this phenomenon which makes it numerically more stable than Euler angles. Moreover, Euler angles and quaternions are more compact and mathematically efficient than the rotation matrices.

Quaternions, introduced by William Hamilton [Hamilton, 1944], are used to encode 3D rotation information into four-element vectors and formulated as $q = w + xi + yj + zk$ where w, x, y and z are scalar values and i, j and k are imaginary numbers. Hence, following equations hold true:

$$i^2 = j^2 = k^2 = -1$$

$$ij = k, \quad jk = i, \quad ki = j, \quad ji = -k, \quad kj = -i, \quad ik = -j$$

$$|q|^2 = w^2 + x^2 + y^2 + z^2$$

$$|q| = 1 \quad \text{(for unit quaternion vectors)}$$

Relationship between rotation matrices and unit quaternions is as follows:

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

Quaternions have a significant role in this thesis because orientation parts of all calibration and other pose values are quaternion vectors. Also, w, x, y, z and $q1, q2, q3, q4$ are used interchangeably in the respective order in both our work and other robotics applications.

3.3 Camera Calibration

Camera calibration, or camera re-sectioning, is one of the most fundamental procedures in computer vision literature. There are two types of camera calibration: computation of internal geometrical and optical attributes of the camera called intrinsic calibration and computation of 3D transformation, i.e. translation and rotation, between the camera and world's or another frame's coordinate system called extrinsic calibration.

Explicit camera calibration is computation of the transformation (extrinsic) parameters between the camera and a reference coordinate system. In vision based robotics applications, extrinsic calibration is the computation of transformation between the camera and the robot. The camera either is fixed at a certain location or attached to the robot and moves with the robot. If the camera is fixed, we need to compute only camera to world coordinate system transformation. However, if the camera is moving along with the robot, two transformations are required which are camera-to-robot and robot-to-world to infer camera-to-world transformation. Camera-robot calibration is also called hand-eye calibration.

Implicit camera calibration estimates the intrinsic (internal) parameters of the lens and image sensor of the camera. It generates a map between camera coordinates and pixel coordinates such as focal length, skew and geometric distortion coefficients. Photogrammetric calibration and self-calibration are the major camera calibration methods. Photogrammetric calibration is done by using a calibration object whose 3D geometric information is known while there is no need prior knowledge for self-calibration. Self-calibration is computation of calibration parameters using a sequence of images taken from a static scene. Photogrammetric calibration is more reliable and precise whereas self-calibration is more flexible. Although implicit camera calibration is a important process to increase accuracy and precision, it is outside the scope of this thesis.

3.4 Neural Network Architectures for Point Clouds

Neural network architectures (used for 3D perception tasks) are divided into two categories differentiated in the use of 3D convolutions.

3D Convolutional Neural Networks: First version of 3D convolutional neural networks uses cuboid kernels with dense representations where empty spaces in point cloud data are represented either as a nil value or the signed distance function [Tchapmi et al., 2017, Dai et al., 2017]. Although this version is intuitive and straight forward along with a wide support by the most popular deep learning frameworks, it is vastly inefficient in terms of processing speed and memory usage for point cloud processing as 3D scans comprised of mostly empty spaces. The second version is sparse convolutional networks [Liu et al., 2015, Graham et al., 2018]. To decrease

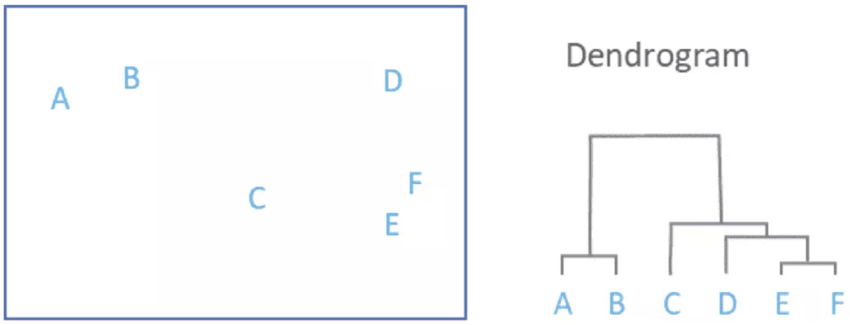


Figure 3.2: Visualization of agglomerative hierarchical clustering [Bock, 2020]

the memory usage and avoid the calculations for empty spaces, the non-empty elements in sparse inputs are stored continuously with their spatial information indexed in a special structure such as a permutohedral lattice or a rectangular grid [Adams et al., 2010, Su et al., 2018].

Neural Networks without 3D Convolutions: In this category of neural network architectures, points are considered feature vectors for multi-layer perceptrons instead of coordinates in the 3D space pioneered by [Qi et al., 2017a]. This method takes fixed size point vectors as input and outputs vectoral predictions for each point.

3.5 Hierarchical Clustering

Clustering, or cluster analysis, is the task of partitioning a set of data entities into clusters (groups) such that entities within the same group are similar to each other while they are distinctively different from the entities in other groups according to some metric such as color difference or distance.

Hierarchical clustering is based on the idea of entities being more similar as they are closer to each other based on a certain connectivity metric such as Euclidean distance. At the beginning of the hierarchical clustering, each entity is considered a separate cluster. Then, following steps are executed:

1. pinpoint two clusters which are closest to each other

2. merge these clusters into one cluster.

These steps are iterated until there isn't any cluster eligible to be merged.

After choosing the connectivity metric, we need to decide our *linkage criteria* which determine how we compute the distances between two clusters. For instance, we can use the longest distance between pair of entities from two clusters (complete-linkage), the distance between centers of two clusters (average-linkage) or the shortest distance between the pair of entities from the clusters (single-linkage). Selection of the linkage criteria heavily depends on the properties of the clustered data. When there isn't sufficient information about the data properties, it is logical to use *Ward's method* [Ward Jr, 1963] that is based on minimization of sum of squared distances of each entity from the average of the entities in the cluster.

Results of a hierarchical clustering algorithm can be visualized on a dendrogram. As shown in Fig. 3.2, we are building clusters in a bottom-up manner which is formally named as agglomerative hierarchical clustering. In contrast to this, we can utilize divisive hierarchical clustering algorithm where we start with one cluster and iteratively divide it into smaller clusters.

3.6 6-DoF Object Pose Estimation

Determination of rotation and translation of objects relative to a reference 3D coordinate system is a long-studied problem in computer vision and robotics. For example, computer vision based robot systems utilize this position and orientation data to manipulate objects or reason about the objects inside the robot's working envelope. Combination of 3D orientation and 3D position information of an object is referred as 6-DoF (6D) object pose. In this thesis, we consider the robot EE an object class and estimate 6-DoF pose of it with two different methods that can be seen in Fig. 5.2 in which red, green and blue arrows respectively represent rotations in x, y and z axes and gray sphere shows the translation of the EE.

6D object pose estimation is usually handled with a single 2D or 3D image, a stereo 2D image pair or a set of sequential images in which the camera is moving and the displacement is known. Generally, objects 6D pose detection solutions work on set of generic objects such as an apple, a pitcher or a driller all of which are available in the YCB dataset [Calli et al., 2017]. Contrarily, 6D pose estimation methods are typically customized for specific objects and struggle to generalize to different object classes.

There are three main techniques to solve 6D object pose estimation problem: with geometric or analytic methods, with evolutionary algorithms and with learning based systems.

3.6.1 Geometric Methods

If shape and geometry of the object is previously known, pose of an object in a given image can be estimated by solving the transformation between descriptors, or keypoints, of a reference object and descriptors of the object in the image. These descriptors can be either calculated with of-the-shelf algorithms such as SIFT, SURF and ORB [Lowe, 2004, Bay et al., 2006, Rublee et al., 2011] or manually defined keypoints such as corners or other distinctive features. Implicit camera calibration parameters are required to calculate 6D object poses in 2D images. It is also possible to estimate the transformation between the reference object point cloud and the target object point cloud in 3D images using registration algorithms such as the ICP algorithm [Besl, 1988] which is explained in Sect. 3.8 where it is not required or possible to predict corresponding descriptors.

3.6.2 Evolutionary Algorithms

Evolutionary algorithms can be used to solve 6D pose estimation problem when online estimation is not necessary. This approach is more powerful than other approaches when camera calibration is imperfect. 6 pose parameters are the genetic parameters in the algorithm and the fitness function is the distance error between projection of the model fitting a set of points and points from the 2D image [Rossi et al., 2005].

3.6.3 Learning Based Systems

These approached utilize machine learning models to learn features to map 2D or 3D inputs to 6D pose estimations. Commonly, these models are trained with a large dataset containing numerous poses of the same object. After training, models are expected to predict 6D poses of the object classes from the training dataset containing unseen poses.

Majority of the latest methods [Xiang et al., 2018, Li et al., 2018b] utilize combination of the approaches explained in this section for 6D pose estimation. They aim to learn transformation between an object instance's point cloud from 3D images and a reference 3D object's CAD model or point cloud which are already provided in public datasets [Calli et al., 2015].

3.7 Keypoint Detection

Keypoint, or descriptor, detection is the task of finding distinctive spatial points of the objects in an image. Keypoint detection plays a crucial role in computer vision tasks where affine transformation between two images needs to be calculated such as panoramic image stitching and real-time localization using 3D sensor data in autonomous driving applications. There are two main ways to extract keypoints from an image: using of-the-shelf generic algorithms to extract keypoints and locating manually defined class-based keypoints.

Generic keypoint detection algorithms like SIFT, SURF and ORB [Lowe, 2004, Bay et al., 2006, Rublee et al., 2011] search for interesting, or feature, points in the image by looking for sudden changes through an image such as edges and corners. Once the interesting points are localized, descriptor vectors are generated for each interesting point. The homogeneous transformation matrix between two images can be created using these keypoints. Correct keypoint-to-keypoint matching between two images is done by using descriptor vectors.

In manual keypoint detection, class-based keypoints are manually set on a reference object/image. For example, we defined total of 6 different keypoints (4 for the corners and 2 for the tip of the grippers) for the EE in this thesis as shown in

Figure 3.3: 2D ICP visualization.

Fig. 5.2. These keypoints are detected using learning techniques or/and analytical computer vision methods during the inference. Since we also defined the classes along with the keypoint locations, classes are also predicted during inference which eliminates the need for descriptor vectors. Affine transformation estimation becomes possible by matching the keypoint locations and classes from two different images as visualized in Fig. 5.3.

Keypoint detection and matching is the conventional way of predicting affine transformation between two frames or object and it has been worked on and utilized in computer vision applications for decades.

3.8 Iterative Closest Point (ICP) Algorithm

Global and local matching of free-form curves and surfaces has been one of the key problems in computer vision. [Besl, 1988] formally introduces the problem as the estimation of the optimal translation and rotation that aligns, or registers, 3D data acquired from the vision sensor(s) describing a 3D shape to a previously available 3D model of the shape in the sensor coordinate system.

The Iterative Closes Point (ICP) registration algorithm has been the foundation of 3D geometric registration in both academic and industrial applications. The inputs are a target point cloud, a reference point cloud and an initial transformation (combination of translation and rotation) which roughly aligns the reference frame to the target frame. The output is the refined transformation which firmly aligns the target and reference point clouds as shown in Fig 3.3.

Essentially, the ICP algorithm iterates over the following steps until either the algorithm converges or maximum number of iterations is reached [Besl and McKay, 1992, Zhou et al., 2018]:

1. Find correspondence set $\mathbf{K} = \{(\mathbf{p},\ \mathbf{r})\}$ from target point cloud $\mathbf{P}$, and reference point cloud $\mathbf{R}$ transformed with current transformation matrix $\mathbf{T}$.

2. Optimize the transformation $\mathbf{T}$ using SVD to minimize Eq. 3.4 calculated over corresponding points in $\mathbf{K}$.

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\ \mathbf{r}) \epsilon K} \|\mathbf{p}\ \textbf{-}\ \mathbf{T} \times \mathbf{r}\|^2 \tag{3.4}$$

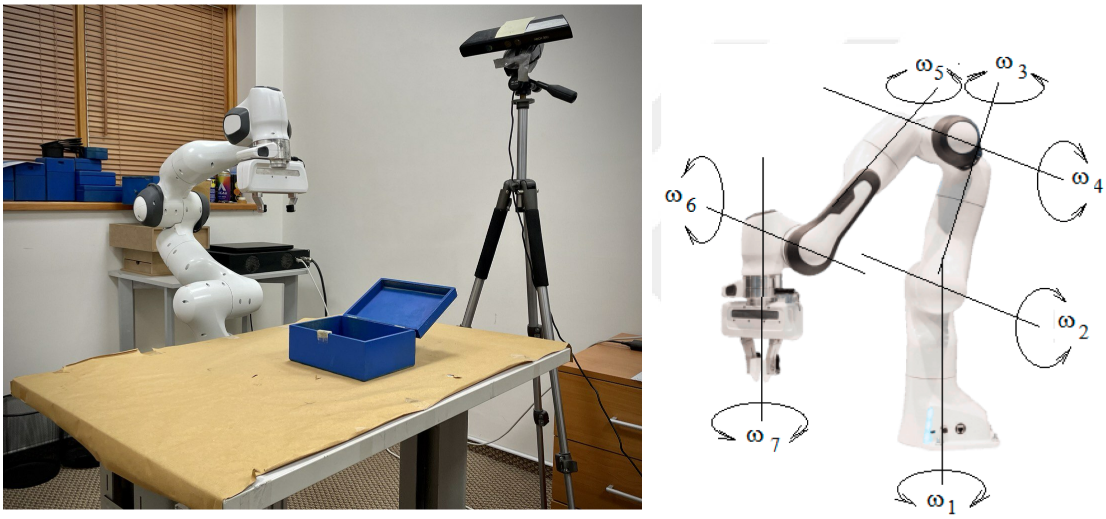
3.9 Conclusion

In this chapter, we provided comprehensive introductions for the fundamental techniques employed in this thesis. We gave details about generic object pose estimation methods, basic concepts of computer vision based robot applications and ways to process point clouds for different needs.

Chapter 4

PRELIMINARIES

4.1 Setup



(a) (b)

Figure 4.1: Setup: (a) outside view of experimental setup with Franka Emika Panda 7-DoF robot arm and Microsoft Kinect V1, (b) joint states visualization of the robot arm.

In this work, we used the Franka Emika Panda 7-DoF Robotic Arm as the calibration target and the Microsoft Kinect V1 depth camera as the vision sensor in the real world setup. The table in front of the robotic arm defines limits the robot's working envelope (operating area) and the depth camera is placed so that it can perceive robot arm in various joint configurations inside the vicinity of the operating area as shown in Fig. 4.1a. Joint states information (Fig. 4.1b) is obtained via ROS frameworks.

4.2 Data Generation

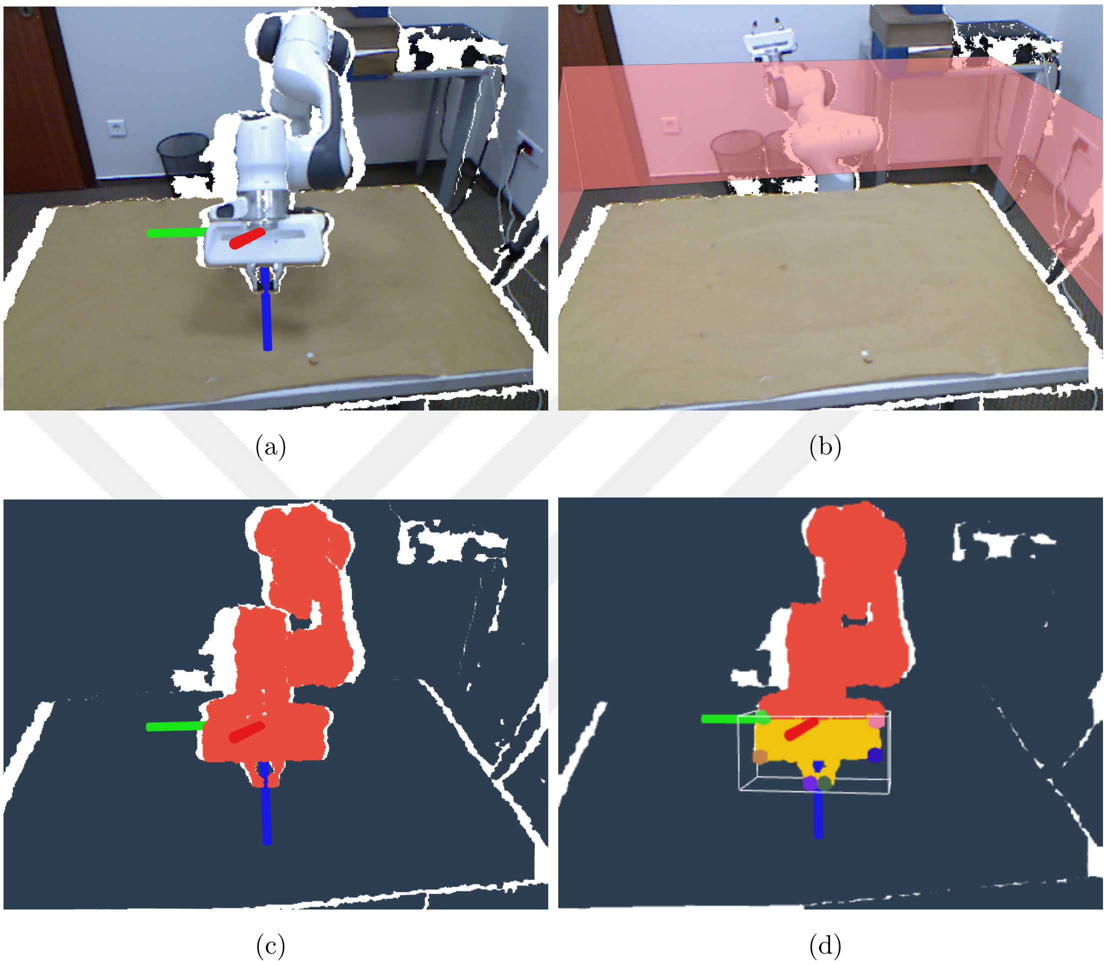


Figure 4.2: Dataset generation stages: (a) points, RGB and EE pose data, (b) background frame and visualization of restricted working envelope (operational area) limits, (c) semantic labels for the robot arm and background, (d) finalized data generation with semantic labels of EE, arm and background; EE pose ground truth and keypoint labels.

Our EE segmentation and pose estimation methods rely on learning as such require a large robot point cloud dataset. We generate ground truth semantic segmentation labels for arm, EE and background; 6-DoF EE poses with respect to the depth camera and 6 keypoint locations and classes of the EE along with the trivially

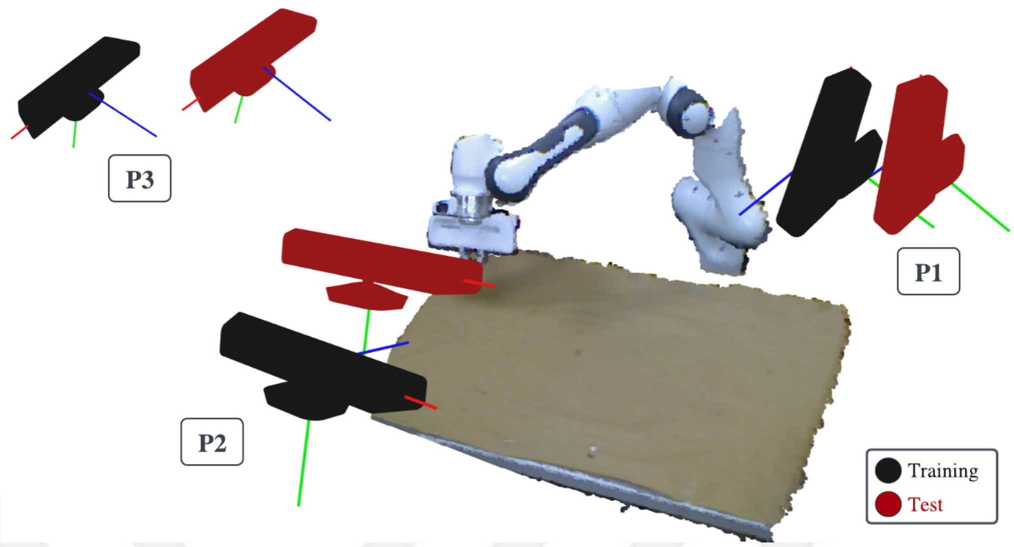


Figure 4.3: Visualization of Kinect camera positions during training and test data collection.

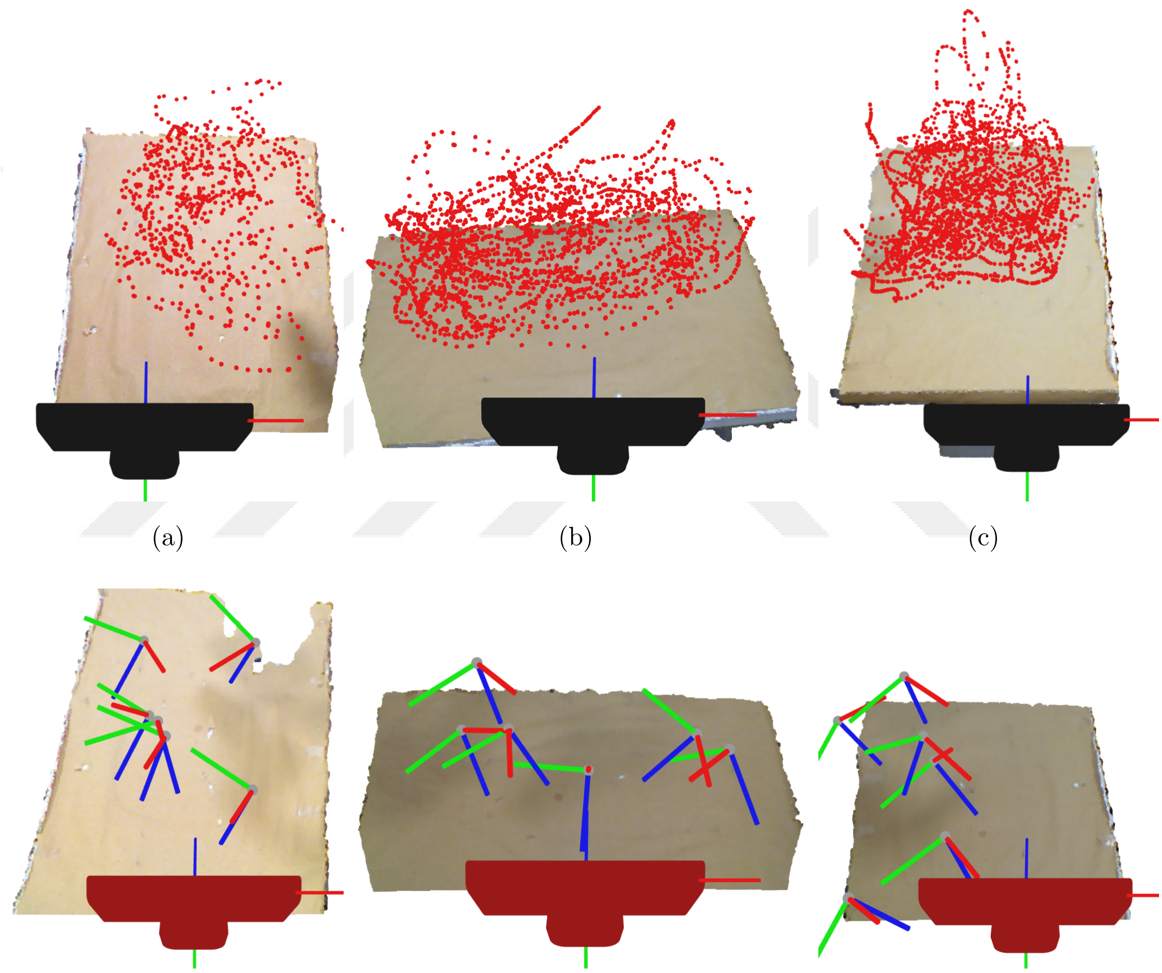
available EE and other joint poses with respect to the robot base.

We automatically collect the specified dataset in a real world setup to eliminate the need to build a simulation environment.

4.2.1 Calibration

The first step is to perform an initial robot-camera calibration. This can be done with existing techniques. Initial calibration is completed with a marker based calibration tool. We execute the first fine-tuning and verification procedure by visualizing that the whole robot arm's CAD model (configured according to current joint states) and the depth camera's point cloud stream tightly overlap using RViz [Kam et al., 2015]. Next step is to match robot CAD model with point cloud stream via ICP algorithm [Zhang, 1994] using current calibration and joint states information. This procedure is the only manual step that is needed for our system and is only required for data collection which is ideally a one-time operation. Once our system is up and running, no such steps will be necessary. Using the calibration information and joint states data, we are able to generate robot's EE pose ground truths with respect to the depth camera. At this stage, we record the point cloud stream with

RGB information, joint states and EE pose data by automatically moving the robot arm to thousands of different joint configurations which are obtained by replaying previously recorded sequential joint states data. Then, we extract frames from the recorded data stream (bag file). A sample point cloud frame from the recorded data is shown in Fig. 4.2a.



(a) (b) (c)

(d) (e) (f)

Figure 4.4: Dataset instances: (a), (b) and (c) show our training dataset collection points for camera positions P1, P2 and P3, respectively. (d), (e) and (f) show our test dataset collection points with EE pose information for camera positions P1, P2 and P3, respectively.

	P1 Test		P2 Test		P3 Test	
	Δ_t [cm]	Δ_R [°]	Δ_t [cm]	Δ_R [°]	Δ_t [cm]	Δ_R [°]
P1 Training	5.15	14	95.12	71.33	173.49	119.07
P2 Training	102.7	96.32	5.88	18.31	88.12	95.37
P3 Training	175.35	170.16	86.44	112.66	6.2	42.5

Table 4.1: Relative translation and rotation differences between camera pose pairs.

4.2.2 Semantic Segmentation Labels

The second step is to automatically generate semantic segmentation labels for EE, arm and background. To achieve that, we first collect background frames from which the robot arm is moved to outside the working envelope which is defined by the edges of the yellow table in our environment. We combine these background frames into a single frame in order to discard transient noise. Next, we specify the physical limits of the robot arm's operating area by defining maximum and minimum x, y, z coordinates with respect to the depth camera as illustrated in Fig. 4.2b. Semantic segmentation labels for the robot arm points and background are generated by subtracting the background points from every frame [Sankoh et al., 2010]. Hence, semantic labeling is finalized for the arm and background as depicted in Fig. 4.2c in which gray points are the background and red points are the robot arm. Further, semantic segmentation labels for EE points are generated by utilizing the robot's joint states, calibration and 3D dimension information of the EE. Using these information, we generate a bounding box with the EE's dimensional data and transform the bounding box using joint states and calibration information as drawn with white lines in Fig. 4.2d. Labels of the arm points inside the transformed bounding box are converted to EE segmentation labels which are colored yellow in Fig. 4.2d. Alternatively, the CAD model of the robot along with joint positions and the calibration information can be used to get the arm and EE points, and consequently background points. This would be needed when the robot arm is

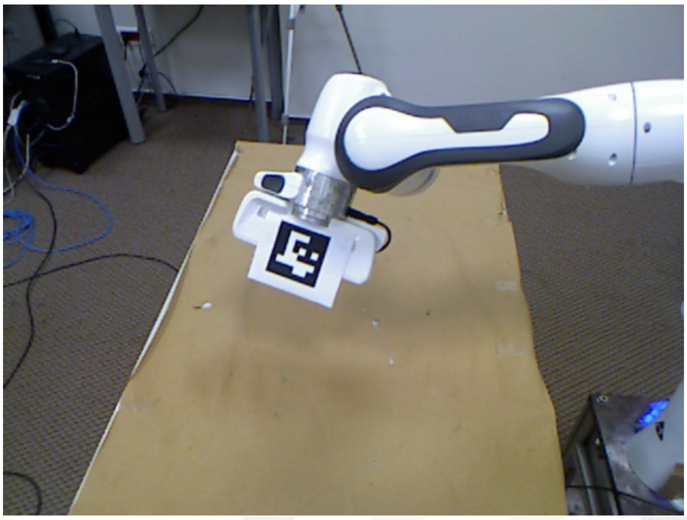


Figure 4.5: AR-tag view.

visible in the frame no matter what, to get the background points.

4.2.3 Keypoint Ground Truth Generation

The third step is to generate keypoint labels for the EE. We extract total of six keypoints; four of them are located at each corner of the EE and two of them are located at the tip of the gripper fingers as shown in Fig. 4.2d and Fig. 5.2 with differently colored hexagons. We define six reference points within the EE bounding box which we already transformed based on the pose information in the previous step. The EE points closest to these reference points are selected as keypoints, as long as their distances are below a certain threshold. If no EE points for a given reference points is within the threshold, we deduce that a keypoint doesn't exist for the corresponding part of the EE. Keypoint classes are generated using the spatial relations of the keypoints.

4.2.4 Dataset Specifications

We collected around 7000 frames from 3 different camera poses (see Fig. 4.4). The cameras are placed on three sides of the robot setup, as shown in Fig. 4.3. We reserve 1000 frames for validation to control overfitting. To increase the generalization, we applied 3D data augmentation methods such as elastic distortion, noise injection and point dropouts during model training. We further collected data from additional 3 cameras poses (see Fig. 4.3), 6 locations per pose and 10 frames per location for a total of 180 frames, to be used as our test data. We show EE poses which we use in our test data in Fig. 4.4 for individual camera poses. The relative translation and rotation differences between training and test camera poses are given in Table 4.1. We also collect evaluation data using an ArUco tag attached to the EE in the same test configuration. We chose a relatively large marker so that it is robust to distance wrt. camera, as shown in Fig. 4.5.

The training data is collected with an initial calibration. As we do not have access to high quality markers (e.g. a motion capture system), this may suffer from all the issues laid out in this paper. The ADD of the training data is calculated as $0.75cm$, which implies a relatively high quality data set. We show data collection points of the EE for each camera position in Fig. 4.4.

4.3 Conclusion

In this chapter, we established the details of the environment and the tools used for this thesis. Further, we introduced a novel way to automatically generate ground truth for data semantic segmentation, EE keypoints and EE pose in typical robot-camera setups.

Chapter 5

SINGLE FRAME END-EFFECTOR POSE ESTIMATION

Our system consists of two main stages; (1) single frame EE 6-DoF pose estimation, and (2) multi-frame extrinsic robot-to-depth camera calibration. The output of the former along with forward kinematics is enough to get a calibration estimate but multiple frames increases robustness. Fig. 5.1 shows our single frame EE pose estimation and Fig. 6.1 shows our calibration workflows. In this chapter, we introduce our approach to the EE pose estimation task (stage (1)). We explain our approach to the robot-camera calibration problem (stage (2)) in Ch. 6.

For EE pose estimation, we first segment the EE from a point cloud using a semantic segmentation approach. Then we utilize two different approaches to estimate the EE pose. The first one does EE rotation prediction followed by a point cloud transformation step to get the EE translation. The second one extracts keypoints from the EE and matches these with reference points calculate the EE pose. Both approaches are followed by an ICP step initialized with the EE pose estimates to match the EE points and the EE CAD model. These steps are summarized in Fig. 5.1. As a result of these steps, we segment and extract EE point cloud from the point cloud frame, predict keypoints of the EE and predict the EE pose shown in Fig. 5.2.

In this chapter, we present the details of our work for single frame EE pose estimation starting from the semantic segmentation step.

5.1 End-Effector Segmentation

The EE pose estimation starts by extracting the semantic segmentation information from an input point cloud. The outputs of this step are the EE, rest of the arm and the background labels for the points as shown in Fig. 1.1 in yellow, red and gray

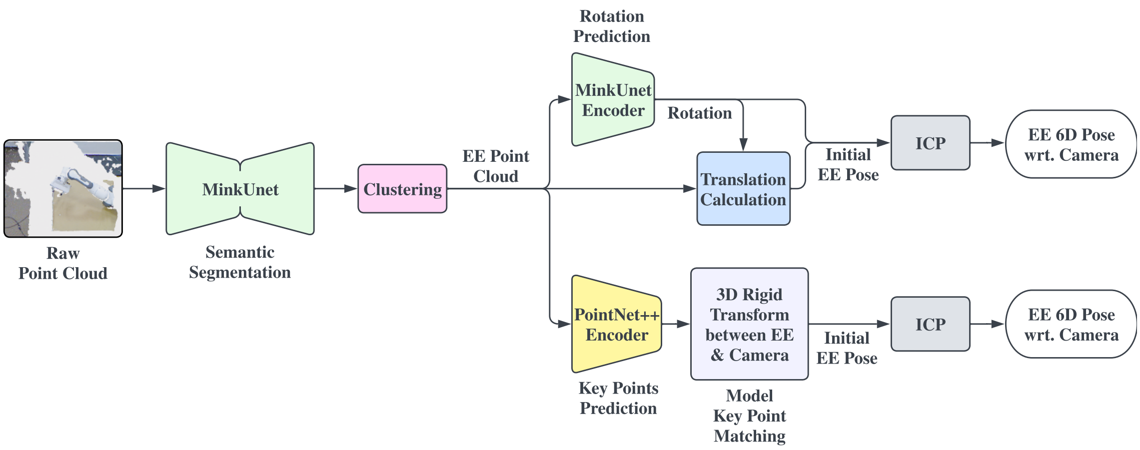


Figure 5.1: Single frame prediction architecture: Starts with semantic segmentation and clustering of EE points, then it is branched into the upper branch which is 6D EE pose estimation with rotation prediction and transform calculation (RPT) and the lower branch which is 6D EE pose estimation via keypoint matching (KPM).

colors for each respective semantic class.

For this step, we utilize a modified, sparse version of the Unet architecture named the Minkowski Unet network [Choy et al., 2019, Liu et al., 2015, Ronneberger et al., 2015]. Unet is one of the most popular neural network backbone architectures to learn spatial relations in images which is crucial for tasks connected to semantic segmentation. We use the full version of MinkUNet18D [Choy et al., 2019] as a backbone for voxel-based feature extraction followed by a two fully connected layers to output the final class predictions. The network is trained setting the voxelized raw point clouds as the input and the collected semantic segmentation data as the target using cross entropy criterion as the objective function. Details of the neural network architecture and training process are shared in Appx. A.1.

We extract the candidate EE points from the input point cloud using these point class predictions. Our network may produce false positive predictions for the semantic classes as any other semantic segmentation architecture might. Although we are not concerned about false positive arm and background predictions, accuracy of the EE semantic predictions is highly important because success of the subsequent methods heavily relies on the quality of the EE point cloud. For example, if the model produces EE classification for both the real EE points and spatially far away non-EE points, then the following systems (shown in Fig.5.1) will not be able to predict the correct results. We apply Ward's agglomerative hierarchical clustering, or linkage clustering, [Ward Jr, 1963] to points classified as EE using the point coordinate values as input to reject superfluous (spatially irrelevant) predictions. We choose the largest cluster and use points in it as the final EE point cloud. This approach can easily be transferred to multi-agent environments by choosing the largest N clusters at the end of the same algorithm where N denotes the maximum number of the agents (robot arms) in a given scene. We do not apply linkage clustering to the other semantic class predictions simply because we do not use the other classes' point clouds in the remaining parts of this system.

5.1.1 EE Expansion Trick

Shape of the EE is roughly a cuboid with two much smaller cylindrical shapes (grippers) attached to it (Fig. 5.2). Two grippers of the EE is prone to be missed during 3D scanning. As a consequence, we are left with fully smooth EE surfaces for prediction that makes our system vulnerable to faulty ICP matching results (Sect. 3.8 and Sect. 5.4). As seen in Fig. 4.2a, the surface becomes uneven, i.e. more descriptive, at the robot wrist where the EE is attached to the robot arm. To benefit from this descriptive surface, we include arm segmentation labels at the wrist to the EE labels during only semantic segmentation training. More specifically, we convert arm points to EE points within the $1.5cm$ periphery of the EE along the negative $z - axis$. We discuss the effects of this trick in detail in Ch. 7.

As a result of this stage, we can produce robust, noise-free EE semantic segmentation outputs.

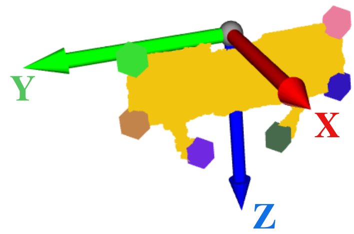


Figure 5.2: Visualization of the end-effector segmentation (yellow), end-effector pose frame, and keypoints (colored hexagons).

5.2 End-Effector Pose Estimation with Rotation Prediction and Transform Calculation: RPT

This approach consists of two main stages; we first predict the EE rotation from the EE points provided by semantic segmentation architecture explained in Sect. 5.1 and calculate the EE translation using rotation prediction and the EE points.

5.2.1 Rotation Prediction

In this part, we train another neural backbone which takes EE point cloud as input and outputs the EE rotation prediction as a quaternion vector (explained in Sect. 3.2). We utilize only the encoder part of another MinkUNet18D [Choy et al., 2019] backbone with the same configuration as the previous one followed by a two fully connected layers to predict 4 dimensional quaternion vector. We opted for a sparse network instead of dense network because voxelization of the EE points makes the rotation predictions faster and more resistant to noise and illumination variations.

Ad-hoc regression of the quaternion vectors with common objective functions such as mean squared loss is not viable due to the nature of the quaternion vectors. We adopted the *pose match loss (PLoss)* from PoseCNN [Xiang et al., 2018] in Eq. 5.1 to learn rotation estimation with high accuracy. It rotates the input point cloud with both predicted rotation and ground truth rotation and measures the average squared distance between rotated points clouds. This loss reaches its minimum value when the predicted rotation and the ground truth rotation are the same. In others words, two rotated point clouds tightly overlap when the loss value is at its minimum.

$$PLoss(\tilde{\mathbf{q}}, \mathbf{q}) = \frac{1}{2m} \sum_{x \epsilon M} \|R(\tilde{\mathbf{q}})\mathbf{x} - R(\mathbf{q})\mathbf{x}\|^2 \tag{5.1}$$

where M denotes the set of EE points, m is the total number of EE points and x is a single point. $\tilde{\mathbf{q}}$ and $\mathbf{q}$ are predicted and ground truth quaternions, respectively. Lastly, $R(\tilde{\mathbf{q}})$ and $R(\mathbf{q})$ are rotation matrices corresponding to $\tilde{\mathbf{q}}$ and $\mathbf{q}$ vectors. Detailed relation between quaternion vectors and rotation matrices are shared in Sect. 3.2.

Network architecture and training details are shared in Appx. A.2.

5.2.2 Translation Calculation

To calculate the EE translation, we use the inverse of the rotation matrix obtained in Sect. 5.2.1 to rotate the points to their canonical locations with respect to the camera. Then we use the relationship between the center of the EE frame and the points to calculate the EE translation in the canonical pose. We finally rotate this back to get the EE translation with respect to the camera.

The Eq. 5.2 is used on each of the EE points to get the canonical locations, where R^{EE} is the rotation prediction of the model, P^{EE} is the position of the EE point and the $\bar{P}^{EE}$ is its rotated position.

$$\bar{P}^{EE} = (R^{EE})^{-1} \times P^{EE} \tag{5.2}$$

When the points are rotated in the camera frame, the relationship between them and the EE frame center is as follows; the x-axis is 0.015m inside the surface of the EE, the y-axis is in the middle and the z-axis is the top of the canonical EE points. The Fig. 5.2 shows these axes superimposed on the segmented EE points. We use the following equations to capture this relationship where $\bar{t}^{EE}$ and $\bar{P}^{EE}$ denote the axes positions of the EE frame and EE points in the rotated frame.

$$\bar{t}^{EE}_x = \max(\bar{P}^{EE}_x) - 0.015 \tag{5.3}$$

$$\bar{t}^{EE}_y = \frac{\max(\bar{P}^{EE}_y) - \min(\bar{P}^{EE}_y)}{2} \tag{5.4}$$

$$\bar{t}^{EE}_z = \min(\bar{P}^{EE}_z) \tag{5.5}$$

We finally rotate $\bar{t}^{EE}$ back to get the EE translation, t^{EE}, wrt. the camera using Eq. 5.6.

$$t^{EE} = R^{EE} \times \bar{t}^{EE} \tag{5.6}$$

Calculating the EE translation with such an approach is efficient and accurate given that the segmentation and rotation predictions are accurate and that the EE is visible. However, full EE visibility is not a given for all the frames and as such, this method cannot be trusted for single frame pose estimation by itself.

As a result of the rotation prediction and translation calculation, we finalize the estimation of EE pose which is show in Fig. 5.2 with red, green, blue arrows corresponding to the rotation and gray sphere corresponding to the translation.

5.3 End-Effector Pose Estimation via Keypoint Matching: KPM

In this approach, we predict keypoints by selecting points from the segmented EE point cloud and match predicted keypoints to reference EE keypoints to figure out pose of the EE with respect to reference frame as shown in Fig. 5.3.

5.3.1 Keypoint Prediction

We start with the definition of 6 hand-picked EE keypoints which are located at four corners of the EE and at the tips of the two gripper fingers of the EE as shown in Fig. 5.2 with differently colored hexagons. Each different color for the keypoints represents a different keypoint class such as top left corner or right gripper. Classification of the keypoints plays a significant role in the simplification and speed of the matching algorithm explained in Sect. 5.3.2. Keypoint estimation is done by selecting points from the EE point cloud itself.

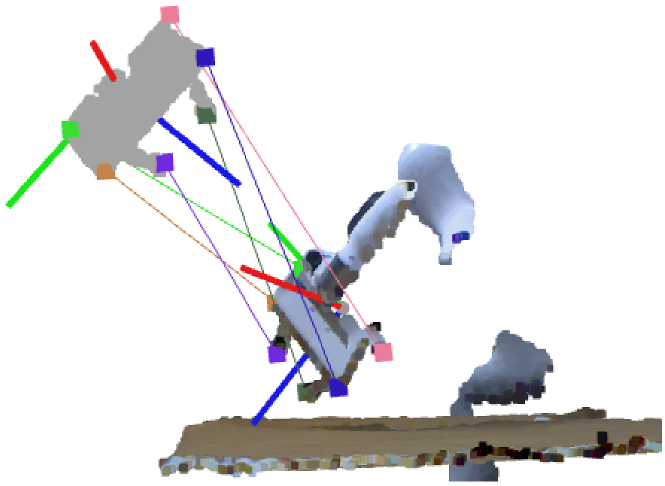


Figure 5.3: Visualization of pose prediction via keypoint-to-keypoint matching between the reference EE and the target EE.

We utilize a dense point cloud encoder PointNet++ [Qi et al., 2017b] as a neural backbone instead of a sparse network because voxelization causes loss of point specific information which prevents precise keypoint selection and prediction. The network outputs 6-class probability estimations for each point in the EE point set. The network is trained with the collected dataset (see Sect. 4.2) using cross entropy loss as the objective function and segmented EE points as input and the collected ground truth keypoints as target. Lastly, we select the points with the highest probabilities for each class and reject the ones whose probabilities are below a predetermined threshold.

5.3.2 Keypoint Matching

We then conduct class-based keypoint matching procedure as shown in Fig. 5.3. We use a version of the least-squares fitting algorithm [Arun et al., 1987] to find the rigid affine transformation between predicted keypoints and the reference keypoints, which gives us the EE pose with respect to the camera in the end. This is possible when there are at least four high quality keypoint predictions because we are working in the 3D space.

5.4 Pose Estimation Refinement via the Iterative Closest Point Algorithm

We described two different techniques to predict the EE pose from segmented EE points; RPT in Sect. 5.2 and KPM in Sect. 5.3. Even though these approaches produce relatively good results, there is room for improvement.

We use an off-the-shelf ICP algorithm [Zhang, 1994, Zhou et al., 2018] to improve the pose estimation accuracy. An ICP algorithm requires three inputs; the initial pose estimate (needs to be fairly accurate), a source point cloud, and a target point cloud as we explained in Sect. 3.8. In our case, the initial pose estimate comes from either the RPT or the KPM methods, and the target point cloud comes from the EE segmentation. To generate the source points, we convert the CAD model of the robot's EE (provided by the manufacturer [Franka Emika, 2022]) to a point cloud, and transform the points based on the current EE configuration. The ICP algorithm outputs the transformation needed to tightly match the source and target points which we use for refinement our initial pose estimates.

5.5 Conclusion

In this chapter, we shared our method for 3-class point cloud segmentation. We introduced two different methods to successfully estimate the 6-DoF pose of the robot EE; RPT and KPM. Finally, we explained how we use an off-the-shelf object registration algorithm to boost EE pose estimation accuracy.

Chapter 6

CAMERA TO ROBOT BASE CALIBRATION

Camera to robot base calibration is an important step in vision based robotics applications because it allows system to be aware of where the EE and other robot joints are located with respect to the camera even if the robot arm is not visible in the camera's field of view as we explained in Sect. 3.1 and Sect. 3.3. In this chapter, we introduce our solution to robot-depth camera calibration task by taking advantage of the methods explained in Ch. 5.

For calibration, the system collects multiple frames and the corresponding EE pose estimates generated in Ch. 5. To increase robustness, sanity check and outlier detection steps are introduced. The remaining poses are then averaged in the rigid-body transformation space (the SE(3) manifold), to get the final extrinsic calibration result. These steps are summarized in Fig. 6.1. In the remainder of the chapter, we explain our methods for sanity check, outlier detection, pose averaging and how we bring together these methods to create a novel camera to robot base extrinsic calibration procedure.

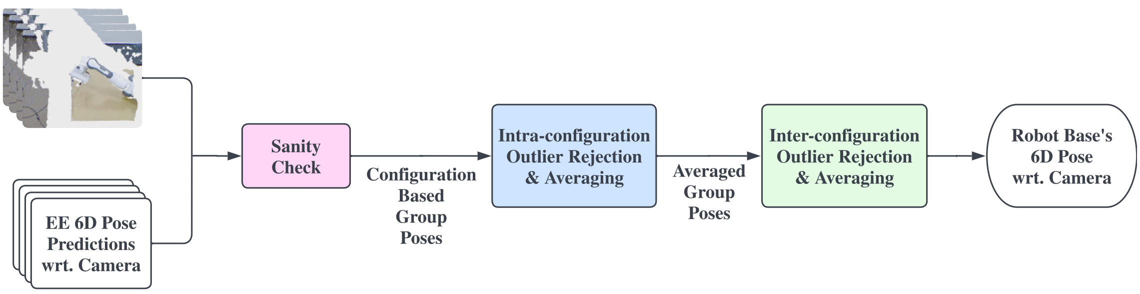


Figure 6.1: Calibration architecture.

6.1 Sanity Check

Sanity checking mechanism is designed to detect tricky EE point clouds and corresponding estimations which would cannibalize the robustness and accuracy of the calibration algorithm. The sanity check algorithm analyzes following conditions;

- if the surface area of the detected EE point cloud is close enough to the real EE surface area,
- if amount of the segmented EE points is more than a certain threshold,
- if the keypoints are in the right positions relative to each other.

In the case that any of the conditions above doesn't hold, we mark the point cloud frame defective in order to prevent it from being used in the calibration process.

6.2 Outlier Detection

Need for outlier detection algorithm arises from the fact that sudden changes in point cloud data and corresponding predictions do happen during the calibration stage even if the camera location and robot joint configuration remain constant because of the transitory illumination variations and other noise sources. Using the point cloud data and estimations for calibration without discarding irregularities would cause inaccuracies in the final extrinsic calibration calculation.

We utilize a modified version of the Z-score outlier detection algorithm using the absolute deviations about the median [Iglewicz and Hoaglin, 1993] to discover the irregular vectors amongst a set of pose vectors. Our pose estimations comprise of translation vectors as x, y, z values in the 3D coordinate system and orientation vectors as quaternion vectors. We directly apply Z-score method to each axis of the translation vector. However, we cannot apply the Z-score method directly to individual elements of the rotation matrices or quaternions. In order to detect outliers amongst rotation predictions, we calculate rotational distances of every prediction to a reference unit quaternion ($q = 1 + 0x + 0y + 0z$). We run the Z-score algorithm on these distances and identify irregular rotation predictions. Lastly, we label a pose vector as outlier if any of the position axes and/or rotation vector is marked outlier.

The outlier detection algorithm takes a set of pose vectors as input and returns a set of pose vectors from which outliers are removed as seen in Algo. 1.

6.3 Pose Averaging

We compute average of the pose vectors for two times in our calibration flow to calculate the final calibration results as shown in Fig. 6.1. A pose vector is a combination of a translation vector and an orientation vector. We calculate the average translation vector by computing the arithmetic average of each axis (x, y, z) individually. However, we cannot compute the average orientation vector in this way due to the structure of rotation matrices and quaternions. In order to compute average orientation, we adopted a quaternion averaging method based on an SO(3) on-manifold optimization technique introduced by [Markley et al., 2007]. Average pose vector is the combination of the average translation and quaternion vectors.

6.4 Calibration Flow

A single EE pose with respect to the camera frame can be used to calculate the transformation between the camera and the robot using Eq. 6.1, where T_X^Y represents the homogeneous transformation between frames X and Y, and the letters B, C, and EE correspond to the robot base, camera and the EE respectively. The EE in the base frame, T_B^{EE}, is obtained by forward kinematics and the EE in the camera frame, T_C^{EE}, is estimated by either the RPT or the KPM methods introduced in Sect. 5.2 and Sect. 5.3, respectively.

$$T_C^B = T_C^{EE} \times (T_B^{EE})^{-1} \tag{6.1}$$

However, a single frame is not robust enough to get a good calibration. We move the robot around and capture multiple frames at each robot pose to get more accurate estimates.

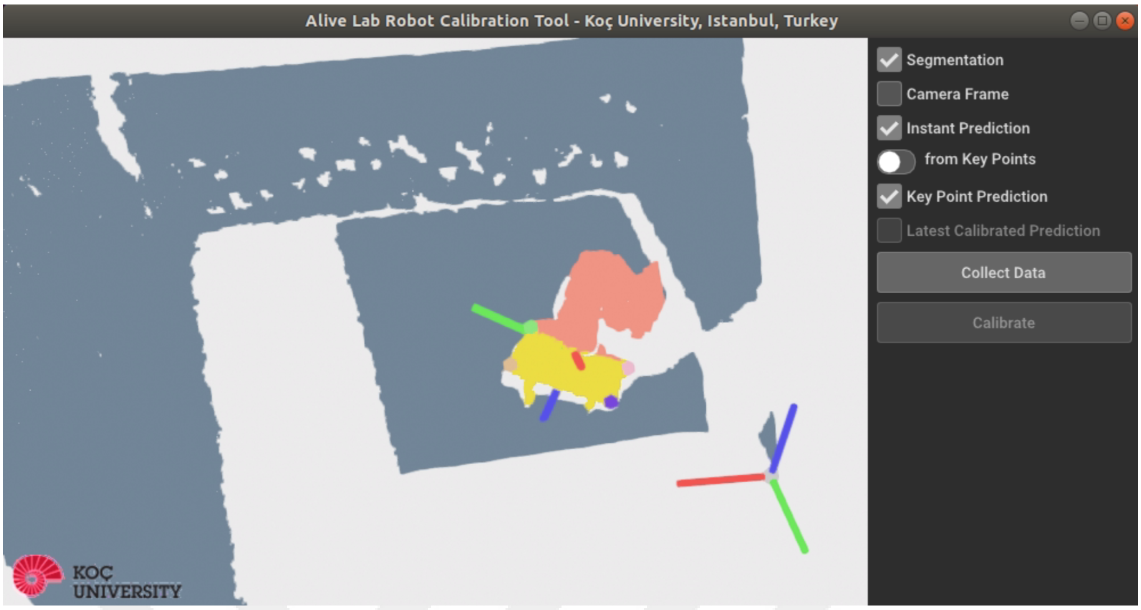


Figure 6.2: Alive Lab Markerless Robot Calibration Tool

We first compute semantic segmentation and EE pose estimates for each point cloud frame as described in Ch. 5. Subsequently, sanity check (Sect. 6.1) is applied to each frame in order to eliminate point cloud frames where the EE is not visible enough for accurate pose estimation.

After the sanity check, the camera to robot base predictions (Eq. 6.1) are computed using the EE poses (Sect. 5.2 and 5.3) and the forward kinematic information. We then group the robot base pose predictions based on the robot configuration (recall that we capture multiple frames per robot configuration). We apply the outlier elimination method (Sect. 6.2) to each robot configuration's pose set in order to discard irregular base poses within the groups. Then remaining camera to base transformations are averaged using the pose averaging technique described in Sect. 6.3. This gives us the individual robot base calibration estimates for each robot configuration group.

Finally, we employ the same outlier detection and averaging steps to the calibration estimates of each robot configuration group to get the final robot calibration estimate. The final pose vector contains ultimate camera to robot base affine transformation information.

6.5 Alive Lab Markerless Robot Calibration Tool

We developed an easy-to-use tool to further simplify the robot-depth camera calibration process by seamlessly integrating the methods introduced in this chapter and Ch. 5 as shown in Fig.6.2. We made the calibration tool publicly available as an open-source project along with the created dataset which has been used for model trainings discussed in this thesis. Users can use this tool for semantic segmentation, keypoint prediction and EE pose estimation via both RPT and KPM in addition to the robot-depth camera calibration. Moreover, users can easily add new features and customize this tool to run their own methods as it is coded in a modular structure.

6.6 Conclusion

In this chapter, we explained our robot-depth camera calibration method. We gave details of our methods to verify the quality of single EE pose predictions, to detect outliers within a group of pose vectors and how to correctly compute mean of pose vector groups. Finally, we unveiled an open-source robot-depth camera calibration software.

Chapter 7

EXPERIMENTS AND RESULTS

In this chapter, we evaluate the semantic segmentation, single pose estimation and camera calibration performances of our system. We test our approach with ground truth segmentation labels as well to gauge the effects of the accuracy of the segmentation model on the pose and calibration estimation. We also provide results without the ICP step to highlight its benefits.

All of our experiments are conducted in a real-life setup which includes a Microsoft Kinect V1 and a Franka Emika Panda robot arm as shown in Fig. 4.1a. For evaluation purposes, we collect a new test set from three previously unseen point of views (POV), using the steps described in Sect. 4.2. We put the robot arm in six different configurations for each POV and collect 10 frames in various lighting conditions. All the results reported in this section uses this test dataset.

We use the translation error (ϵ_t), rotational error (ϵ_R) and the average distance (ADD) metrics to measure the pose estimation and calibration performances. The ϵ_t is the Euclidean distance between the ground truth translation and predicted translation. The ϵ_R is the minimum rotation in degrees between the ground truth rotation and the predicted rotation. ADD is the average of point-to-point Euclidean distances between the EE points transformed with the ground truth pose and transformed with the predicted pose [Hinterstoisser et al., 2012]. ADD is the acknowledged way of computing object pose prediction errors which considers both translation and rotation. To measure the accuracy of our semantic segmentation model, we use the standard accuracy, precision and recall metrics.

Class	Precision	Recall	Accuracy
End-Effector	0.98	0.90	1.00
Arm	0.90	1.00	0.99
Background	1.00	1.00	1.00
Overall	0.96	0.96	0.99

Table 7.1: Semantic segmentation results.

Class	Precision	Recall	Accuracy
End-Effector	0.96	0.99	1.00
Arm	0.87	0.99	0.99
Background	1.00	0.99	0.99
Overall	0.94	0.99	0.99

Table 7.2: Semantic segmentation results with the EE expansion trick (Sect. 5.1.1).

7.1 Semantic Segmentation

Semantic segmentation performance is important since it is the first step and its EE segmentation output is used as input to multiple the succeeding steps. We use the per-class precision and recall statistics along with the overall accuracy to assess the performance of segmentation model which is described in Sect. 5.1.

Table 7.1 and Table 7.2 present the semantic segmentation results differing based on the EE expansion. As explained in Sect. 5.1.1, we expand limits of our EE labels in order to enhance the final pose and calibration predictions. The trick allows the segmentation model to find all of the EE points while decreasing precision. Therefore, we are able to feed all the EE points to our models. Our main aim with the trick is to input more descriptive surfaces to the ICP algorithm for better EE pose estimation, hence better calibration estimation. Overall, the segmentation

Method	ϵ_t [cm]	ϵ_R [°]	ADD [cm]
RPT	2.36 ± 0.36	7.75 ± 5.05	2.47 ± 0.52
RPT + ICP	0.87 ± 0.20	3.20 ± 2.42	0.97 ± 0.30
KPM	1.35 ± 0.37	7.46 ± 1.54	1.53 ± 0.33
KPM + ICP	0.96 ± 0.27	2.42 ± 0.98	0.99 ± 0.26

Table 7.3: End-effector pose estimation results with ground truth semantic segmentation.

model performs well. The recall ratio for EE segmentation is $\sim$ 100%, i.e., the model is able to catch all the EE points. Recall that we use linkage clustering to get rid of spurious segmentation results to improve down-stream performance. As expected, only translation estimation part of the lean RPT method is negatively affected by the EE expansion due to the calculation technique of the translation via RPT (Sect. 5.2.2).

The pose estimation and extrinsic calibration performances are calculated with both the model outputs (Table 7.4) and ground truth segmentation labels (Table 7.3). They show that there is around $0.05cm$ translation and $0.5°$ rotation difference between the average errors, with less variance on the ground truth end. These are negligible when the camera noise is taken into account [Wasenmüller and Stricker, 2016]. **Combining this with the segmentation performance, we conclude that the segmentation model performs well enough to generate reliable inputs for the subsequent modules.**

7.2 Single Frame Pose Prediction and Multi-Frame Calibration

Table 7.4 and Table 7.5 show the pose estimation performances of the RPT and KPM with and without the ICP post-processing step with semantic segmentation predictions with and without the expansion trick, respectively. The Table 7.3 shows the same with ground truth segmentation. Table 7.6 shows the calibration results.

Method	ϵ_t [cm]	ϵ_R [°]	ADD [cm]
RPT	2.49 ± 0.77	7.18 ± 4.62	2.58 ± 0.90
RPT + ICP	1.01 ± 0.32	3.47 ± 2.75	1.07 ± 0.35
KPM	1.43 ± 0.48	7.44 ± 1.44	1.60 ± 0.47
KPM + ICP	$\mathbf{1.00 \pm 0.27}$	$\mathbf{2.74 \pm 1.59}$	$\mathbf{1.04 \pm 0.27}$

Table 7.4: End-effector pose estimation results with semantic segmentation predictions using the EE expansion trick (Sect. 5.1.1).

Method	ϵ_t [cm]	ϵ_R [°]	ADD [cm]
RPT	2.03 ± 0.59	7.04 ± 3.51	2.15 ± 0.55
RPT + ICP	1.16 ± 0.40	2.83 ± 1.20	1.20 ± 0.40
KPM	1.54 ± 0.45	8.10 ± 2.11	1.75 ± 0.43
KPM + ICP	1.15 ± 0.40	2.85 ± 1.14	1.19 ± 0.40

Table 7.5: End-effector pose estimation results with semantic segmentation predictions.

ICP step provides considerable improvement for pose estimation: The addition of the ICP post-processing step improves both algorithms in all metrics in both segmentation input cases. In addition, the absolute performance is very impressive with about $1cm$ translation and 3° rotation errors. This would be satisfactory for an important amount of manipulation applications with grippers.

KPM has a slight edge over RPT: Both methods perform more or less the same with the ICP step. KPM outperforms RPT in translation estimation and RPT outperforms KPM in rotation estimation when we remove the ICP step.

Pose Estimation Comparison to Existing Work: It is difficult to compare the methods since the setups are significantly different, mainly due to simulation

	ϵ_t [cm]	ϵ_R [°]
AR-Tag	1.83	3.26
Ours	2.35	3.37
Ours + ICP	0.74	1.69
GT + ICP	0.77	1.19

Table 7.6: Calibration results for different methods.

requirements and lack of point cloud data in existing datasets. However, we are going to attempt comparison wherever there are similarities while conceding that our analysis is not apples to apples.

RPT and KPM approaches even without the ICP step outperform state-of-the-art object pose detection methods [Xiang et al., 2018, Li et al., 2018b] having $2.12cm$ and $1.54cm$ average ADDs, respectively. In other words, these modules predicts EE pose with 100% accuracy in the case that ADD threshold is chosen above approximately $1.07 + 0.35 = 1.42cm$ for RPT and $1.04 + 0.27 = 1.31cm$ for KPM as shown in Table 7.5. The performance gets much better when ICP step is included. Moreover, we outperform a similar single frame pose estimation method [Lee et al., 2020] whose maximum ADD rate is below 60% for Kinect V1. However, the generic object pose detection methods deal with multiple objects [Calli et al., 2015] and [Lee et al., 2020]'s result is for the entire robot arm, whereas we are only looking at the EE.

[Valassakis et al., 2021] report ϵ_t $(1.34cm)$ and ϵ_R $(4.4°)$ errors on their simulation data and ϵ_t $(1.04cm)$ on their real world data, for their eye-in-hand calibration method. Our ϵ_t and ϵ_R errors are lower; however there is a significant setup discrepancy between our methods (eye-to-hand vs. eye-in-hand).

High Performance Multi-Frame Calibration: The calibration results are given in Table 7.6. The average translation and rotational error between the predicted calibration and the ground truth calibration are $\epsilon_t = 0.74cm$ and $\epsilon_R = 1.69°$, respectively. As expected, the ICP step significantly improves performance and that

	P1 Test		P2 Test		P3 Test	
	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]
P1 Training	1.32	0.64	1.64	2.81	4.21	5.63
P2 Training	0.84	0.60	0.70	1.82	1.61	2.32
P3 Training	1.82	1.20	2.28	2.12	0.93	1.89

Table 7.7: Calibration prediction results for respective training and test sets.

semantic segmentation and ground truth inputs perform similarly. This absolute performance would allow for most manipulation applications outside of assembly or high precision tasks. These results are affected by the sensor noise, and potentially the intrinsic calibration and the initial extrinsic calibration errors. We argue that a more recent camera or an active marker system is needed to go beyond these values.

We also compare our system with AR-tag based classical baseline. The AR-tag results in Tab. 7.6 are given without the ICP step. These results show that the AR-tag holds $0.5cm$ advantage in translation and is on par in rotation. We chose a relatively large marker to be robust to distance and large orientation changes. This alters the EE shape considerably, shown in Fig. 4.5, and makes ICP infeasible. A smaller tag degrades the performance considerably with distance to camera. We argue that, if we were to have an appropriate reference CAD model with the large marker, the final calibration results would be on-par. Thus, our method can achieve similar results to a classical marker based method while being easier to use and requiring much less manual effort down the line. We further argue that our system would have better performance than an AR-tag based approach with a smaller marker that is more practical to use.

7.2.1 Training with Data from Individual Camera Poses

The results we presented so far were obtained from data collected from each training camera location (see Fig. 4.3). These locations cover either side and the front of the robot. They also cover range of distances to robot base where P1 is closest and P3 is the farthest. However, this raises the question of calibration performance without such coverage.

To answer this, we train different models using data from each training camera location and test it on all the test data (e.g. train with P2-training, test on P1-test, P2-test and P3-test). As given in Table. 4.1, there are significant translation and rotation differences between the poses with only similarity being the pitch axis.

Table 7.7 shows the calibration results. The calibration results show that, the closer the train-test locations, the better the performance with an exception for P1-train and P1-test. The reason is that the P1-train pose is very close to setup, and as such, its training set lacks data from farther EE poses. System trained with P2 data achieves the best overall results, as expected, since it is in the middle both the translation-wise and the rotation-wise. Furthermore, its performance is not too far from the system trained with all the data.

Table 7.8 and Table 7.9 show our EE pose estimation results for RPT + ICP and KPM + ICP, respectively. In both tables, P1 test set results are very close to each other regardless of the training dataset. For P2 and P3 test sets, P1 training set falls behind while the results from P2 and P3 training sets are vastly close to each other with approximately $1mm$ translation and $1.5°$ rotation differences. It implies that training sets where the camera is closer to the robot base struggle to predict EE poses at further camera positions. Contrarily, training datasets with distant camera positions are able to produce more accurate estimations at closer camera positions. In training datasets, average number of points in EE clouds is directly proportional to the distance between the camera and the robot base. Consequently, we argue that models trained with smaller EE point clouds are able to fit larger EE point clouds whereas the opposite is not true.

The EE pose estimation results is not directly reflected on the calibration results. This is due to the sanity check and outlier elimination procedures explained in Sect. 6.1 and Sect. 6.2, respectively. Although the average pose estimations for P2 and P3 are very similar, predictions of the models trained on P3 set have more deviation in rotation estimations than P2 set that deeply affects the calibration results. Average standard deviations for rotation predictions are 5.62°, 1.26° and

	P1 Test		P2 Test		P3 Test	
	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]
P1 Training	0.62	2.99	1.36	6.08	2.10	9.91
P2 Training	0.70	2.97	1.07	2.82	1.47	3.43
P3 Training	0.72	2.50	0.94	4.09	1.26	3.34

Table 7.8: RPT + ICP EE pose prediction results for respective training and test sets.

	P1 Test		P2 Test		P3 Test	
	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]	ϵ_t [cm]	ϵ_R [degree]
P1 Training	0.67	6.59	1.46	6.94	1.82	6.99
P2 Training	0.66	2.87	1.08	2.80	1.50	2.96
P3 Training	0.65	2.87	1.04	7.82	1.35	2.20

Table 7.9: KPM+ ICP EE pose prediction results for respective training and test sets.

4.80° for models trained on P1, P2 and P3 datasets, respectively.

The results imply that, a careful selection of the training camera pose is important but a single camera is enough with a slight sacrifice in calibration performance.

7.3 Conclusion

In this section, we investigated how our segmentation model performs and its performance affects subsequent modules. We also explained how the EE expansion trick effects the segmentation and following modules. Further, we showed how two totally different approaches (RPT and KPM) can perform on EE pose estimation and how much an off-the-self ICP algorithm enhances the both results. We examined our robot-depth camera calibration results. Finally, we discussed how different camera poses affect the EE pose estimations and robot-camera calibration estimations.

Chapter 8

CONCLUSION AND FUTURE WORK

In this work, we presented a learning based extrinsic calibration system that does not require fiducial markers, additional hardware or simulation. Our system takes point cloud images, segments out the end-effector (EE), feeds these EE points to two single-frame EE pose estimation approaches that use learned models and the ICP algorithm, and finally calculates the robot-camera extrinsic calibration parameters from multiple EE pose predictions. The training data is collected automatically from the real robot setup with initial calibration being the only needed human intervention.

In automatic data collection stage, we record point clouds and kinematics information for different robot joint positions. Then using the calibration information and forward kinematics, we automatically generate ground truth data for EE pose, semantic segmentation labels and EE keypoints for the point clouds. EE keypoints are generated for manually selected EE parts.

In single frame EE pose estimation stage, we train a neural network to predict semantic segmentation labels for point clouds. We refine EE segmentation results by applying linkage clustering to raw EE segmentation results. We estimate EE pose via two different methods (RPT and KPM) using the EE point cloud. In RPT, we train a encoder neural network to predict EE rotation, then using the EE point cloud and the rotation prediction we predict the EE translation to complete EE pose prediction. In KPM, we train an MLP-based encoder neural backbone to predict six keypoints for the EE point cloud. We compute the EE pose using the predicted keypoints and reference keypoints. Finally, we enhance our pose predictions by applying the ICP algorithm to both RPT and KPM outputs.

In the extrinsic calibration stage, we calculate camera to robot base pose using a set of single frame EE predictions from different robot positions and forward kinematics. We reject unreliable EE pose prediction in the sanity check step. Using the reliable estimations and forward kinematics we calculate average robot base poses for each robot arm position after eliminating outlier robot base poses. Lastly, we calculate final camera to robot base extrinsic calibration by averaging all robot base pose following outlier elimination.

Our system achieves sub-centimeter level position and sub-deciradian level rotation precision with our setup. We argue that standard augmented reality tag markers and our depth camera combination would not have achieved better results especially with the inclusion of the ICP algorithm. We further argue that less noisy depth cameras and higher precision robot arms would result in even better performance.

8.1 Future Work

There are potential future steps to expand and improve our system. A related assumption in this vein is that our system assumes that there are no objects in the camera view during calibration since we do not collect data for them as adding object segmentation to data collection would compromise the data collection. We argue that this is a mild requirement, to remove all the objects from the environment, to handle extrinsic calibration since it can be done prior to any application.

A direct future step to alleviate the aforementioned assumption is to be able to segment arbitrary objects. There is interest in the robotics field for this problem and some early approaches that are not widely applicable yet. Data augmentation techniques and/or further developments in deep learning may make this feasible. An augmentation example would be to superimpose object point clouds from available point cloud object datasets to train an object class for semantic segmentation.

We use a relatively high-capacity model, MinkUnet18D [Choy et al., 2019], for the semantic segmentation task involving only three classes. As such, it is prone to overfitting. An out of sample input (i.e. a point cloud with objects) would be difficult to handle. However, using lower capacity models such as MinkUNet101 and MinkUNet14A deteriorated our semantic segmentation performance. The object data injection would also be helpful as a regularizer for our high capacity model.

Our system segments the background and the robot body without the EE. These can be leveraged for other tasks or make single-frame EE pose estimation more robust, e.g., by estimating the joint angles from the robot points. If arbitrary object segmentation capability is also introduced, this system would evolve to be highly beneficial for manipulation tasks.

Currently, we provide the desired keypoints by hand. Even though it is enough to do this once for each EE, it is not the most ideal approach since the user may not be familiar enough with robotics to do so or the selected keypoints may not be easy to distinguish. An automated keypoint selection algorithm can be added in the future to handle this challenge.

We collect data with the robot arm in different joint configurations for the calibration operation. These joint configurations should be diverse for a better calibration estimate. The test set in this work, which would be the calibration set in real operation, randomly selected the joint configurations. A fully automated system should chose these by itself while keeping the EE in the camera frame.

8.2 Concluding Remarks

Our system is a hybrid of learning based and classical approaches. We take advantage of the recent developments in deep learning and combine the model outputs with the ICP algorithm, sanity check and outlier rejection steps. The noisy data and inherent errors in these models result in reduced performance which are offset by the classical steps. On the other hand, ICP does not perform well from arbitrary starting points and is helped immensely from initial model predictions. We think that hybrid approaches are required for robotic tasks where labelled data is expensive and time consuming to produce and there is already suitable underlying geometry/physics which we can leverage.

BIBLIOGRAPHY

[Adams et al., 2010] Adams, A., Baek, J., and Davis, M. A. (2010). Fast high-dimensional filtering using the permutohedral lattice. In *Computer graphics forum*, volume 29, pages 753–762. Wiley Online Library.

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

[Besl, 1988] Besl, P. J. (1988). Geometric modeling and computer vision. *Proceedings of the IEEE*, 76(8):936–958.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie.

[Bock, 2020] Bock, T. (2020). What is Hierarchical Clustering? . `https://www.displayr.com/what-is-hierarchical-clustering/`. [Online; accessed 19-July-2022].

[Calli et al., 2017] Calli, B., Singh, A., Bruce, J., Walsman, A., Konolige, K., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2017). Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268.

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

[Cheng et al., 2021] Cheng, R., Razani, R., Taghavi, E., Li, E., and Liu, B. (2021). 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12547–12556.

[Choy et al., 2019] Choy, C., Gwak, J., and Savarese, S. (2019). 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084.

[Dai et al., 2017] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

[Deng et al., 2012] Deng, S., Fang, D., Cai, Z., Liao, H., and Montavon, G. (2012). Application of external axis in robot-assisted thermal spraying. *Journal of thermal spray technology*, 21(6):1203–1215.

[Fiala, 2005] Fiala, M. (2005). Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE.

[Franka Emika, 2022] Franka Emika (2022). Franka Emika ROS. `https://github.com/frankaemika/libfranka`. [Online; accessed 9-June-2022].

[Garrido-Jurado et al., 2014] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292.

[Graham et al., 2018] Graham, B., Engelcke, M., and Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232.

[Hamilton, 1944] Hamilton, W. R. (1944). Quaternions. In *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*, volume 50, pages 89–92. JSTOR.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[Hinterstoisser et al., 2012] Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., and Navab, N. (2012). 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.

[Huber et al., 2016] Huber, P., Hu, G., Tena, R., Mortazavian, P., Koppen, P., Christmas, W. J., Ratsch, M., and Kittler, J. (2016). A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th international joint conference on computer vision, imaging and computer graphics theory and applications*.

[Hwang et al., 2020] Hwang, M., Thananjeyan, B., Paradis, S., Seita, D., Ichnowski, J., Fer, D., Low, T., and Goldberg, K. (2020). Efficiently calibrating cable-driven surgical robots with rgbd fiducial sensing and recurrent neural networks. *IEEE Robotics and Automation Letters*, 5(4):5937–5944.

[Iglewicz and Hoaglin, 1993] Iglewicz, B. and Hoaglin, D. (1993). *Volume 16: how to detect and handle outliers, The ASQC basic references in quality control: statistical techniques, Edward F. Mykytka*. PhD thesis, Ph. D., Editor.

[Jakab et al., 2021] Jakab, T., Tucker, R., Makadia, A., Wu, J., Snavely, N., and Kanazawa, A. (2021). Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12783–12792.

[Jiang et al., 2020] Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.-W., and Jia, J. (2020). Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition*, pages 4867–4876.

[Kam et al., 2015] Kam, H. R., Lee, S.-H., Park, T., and Kim, C.-H. (2015). Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345.

[Kazemi and Sullivan, 2014] Kazemi, V. and Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1867–1874.

[Kroeger et al., 2019] Kroeger, O., Huegle, J., and Niebuhr, C. A. (2019). An automatic calibration approach for a multi-camera-robot system. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1515–1518. IEEE.

[Labbé et al., 2021] Labbé, Y., Carpentier, J., Aubry, M., and Sivic, J. (2021). Single-view robot pose and joint angle estimation via render & compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1654–1663.

[Lambrecht, 2019] Lambrecht, J. (2019). Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection. In *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)*, pages 136–141. IEEE.

[Lee et al., 2020] Lee, T. E., Tremblay, J., To, T., Cheng, J., Mosier, T., Kroemer, O., Fox, D., and Birchfield, S. (2020). Camera-to-robot pose estimation from a single image. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9426–9432. IEEE.

[Li et al., 2018a] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018a). Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31.

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

[Liu et al., 2015] Liu, B., Wang, M., Foroosh, H., Tappen, M., and Pensky, M. (2015). Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–814.

[Liu et al., 2020] Liu, X., Madhusudanan, H., Chen, W., Li, D., Ge, J., Ru, C., and Sun, Y. (2020). Fast eye-in-hand 3-d scanner-robot calibration for low stitching errors. *IEEE Transactions on Industrial Electronics*, 68(9):8422–8432.

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

[Markley et al., 2007] Markley, F. L., Cheng, Y., Crassidis, J. L., and Oshman, Y. (2007). Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197.

[Olson, 2011] Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE.

[Pauwels and Kragic, 2016] Pauwels, K. and Kragic, D. (2016). Integrated on-line robot-camera calibration and object pose estimation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2332–2339. IEEE.

[Peng et al., 2020] Peng, H., Yang, X., Su, Y.-H., and Hannaford, B. (2020). Real-time data driven precision estimator for raven-ii surgical robot end effector position. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 350–356. IEEE.

[Qi et al., 2017a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). 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.

[Qi et al., 2017b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[Rossi et al., 2005] Rossi, C., Abderrahim, M., and Diaz, J. C. (2005). Evopose: A model-based pose estimation algorithm with correspondences determination. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 3, pages 1551–1556. IEEE.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520.

[Sankoh et al., 2010] Sankoh, H., Ishikawa, A., Naito, S., and Sakazawa, S. (2010). Robust background subtraction method based on 3d model projections with likelihood. In *2010 IEEE International Workshop on Multimedia Signal Processing*, pages 171–176. IEEE.

[Shiu and Ahmad, 1987] Shiu, Y. C. and Ahmad, S. (1987). Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb. *Journal of Intelligent and Robotic Systems*.

[Su et al., 2018] Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539.

[Tchapmi et al., 2017] Tchapmi, L., Choy, C., Armeni, I., Gwak, J., and Savarese, S. (2017). Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE.

[Toshev and Szegedy, 2014] Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660.

[Tsai and Lenz, 1988] Tsai, R. Y. and Lenz, R. K. (1988). A new technique for fully autonomous and efficient 3d robotics hand-eye calibration. In *Proceedings of the 4th international symposium on Robotics Research*, pages 287–297.

[Valassakis et al., 2021] Valassakis, E., Dreczkowski, K., and Johns, E. (2021). Learning eye-in-hand camera calibration from a single image. *CoRR*, abs/2111.01245.

[Vu et al., 2022] Vu, T., Kim, K., Luu, T. M., Nguyen, T., and Yoo, C. D. (2022). Softgroup for 3d instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2708–2717.

[Ward Jr, 1963] Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

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

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

[You et al., 2020] You, Y., Lou, Y., Li, C., Cheng, Z., Li, L., Ma, L., Lu, C., and Wang, W. (2020). Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13647–13656.

[Zhan and Wang, 2012] Zhan, Q. and Wang, X. (2012). Hand–eye calibration and positioning for a robot drilling system. *The International Journal of Advanced Manufacturing Technology*, 61(5):691–701.

[Zhang, 1994] Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152.

[Zhou et al., 2018] Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847.*

Appendix A

MODELS AND TRAINING DETAILS

A.1 Semantic Segmentation

We use whole Minkows Unet for semantic segmentation learning as shown in Fig.A.1 followed by two fully connected layers whose hidden layers sized 256 and 1024. MinkUnet18D has the following plane count configuration: 32, 64, 128, 256, 384, 384, 384, 384, 384.

Model training details:

- Learning rate: 0.00001
- Batch size: 2
- Voxel size: 0.005
- Epoch: 15
- Augmentation: elastic, noise, transform, flip, gravity
- numpy & pytorch seed: 1

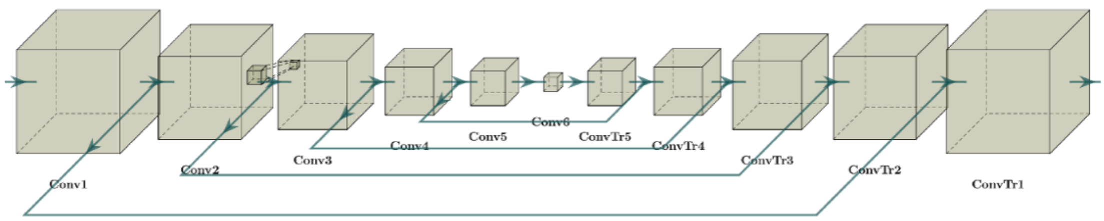


Figure A.1: Minkowski UNet.

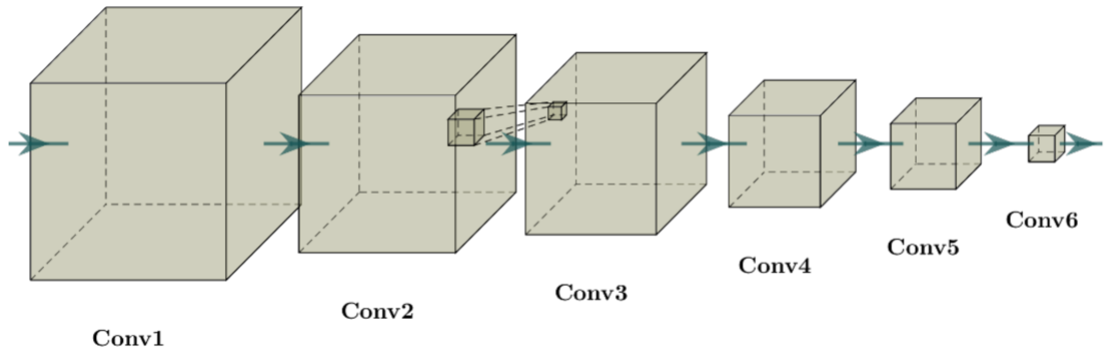


Figure A.2: Minkowski UNet encoder.

A.2 Rotation Prediction

We use Minkows Unet encoder for rotation learning as shown in Fig.A.2 followed by two fully connected layers whose hidden layers sized 2048 and 2048. MinkUnet18D encoder has the following plane count configuration: 32, 64, 128, 256, 384

Model training details:

- Learning rate: 0.00001
- Batch size: 64
- Voxel size: 0.005
- Epoch: 96
- Augmentation: noise, gravity
- numpy & pytorch seed: 1

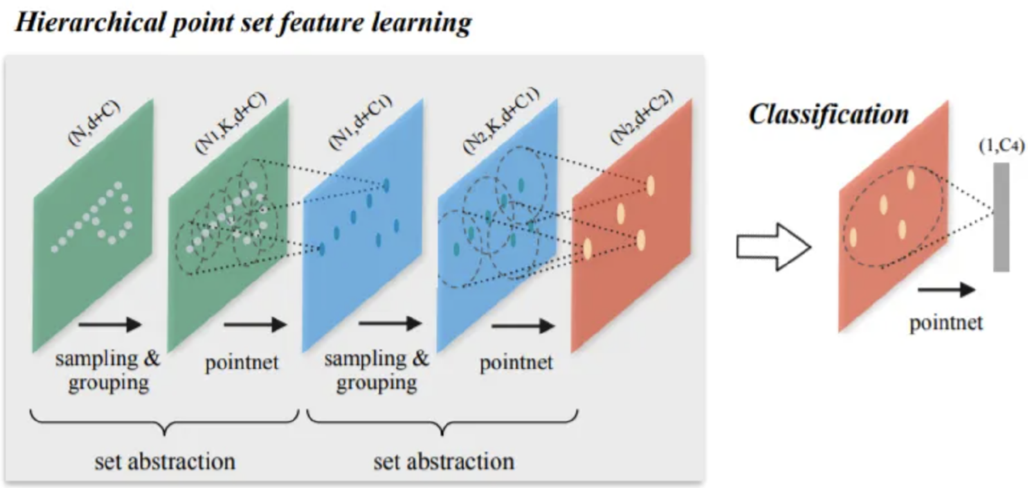


Figure A.3: Pointnet++ encoder.

A.3 Keypoint Prediction

We use Pointnet++ encoder for rotation learning as shown in Fig.A.3.

Model training details:

- Learning rate: 0.00001
- Batch size: 32
- Epoch: 114
- Augmentation: elastic, noise, transform, flip, gravity
- numpy & pytorch seed: 1

Appendix B

OUTLIER REMOVAL ALGORITHM

Algorithm 1: Outlier removal algorithm

```
Input: poseSet                                    // set of pose vectors
Output: prunedPoseSet   // set of pose vectors with no outliers
translationsX = poseSet[:, 0]
translationsY = poseSet[:, 1]
translationsZ = poseSet[:, 2]
rotations = poseSet[:, 3:7]
unitQuaternion = [1, 0, 0, 0]
rotationalDistances = emptyList

for r in rotations do
    rd = computeRotationalDistance(r, unitQuaternion)
    append rd to rotationalDistances

nonOutlierIndicesX = removeOutlierIndices(translationsX)
nonOutlierIndicesY = removeOutlierIndices(translationsY)
nonOutlierIndicesZ = removeOutlierIndices(translationsZ)
nonOutlierIndicesQ = removeOutlierIndices(rotationalDistances)
nonOutlierIndicesAll = nonOutlierIndicesX ∩ nonOutlierIndicesY ∩
    nonOutlierIndicesZ ∩ nonOutlierIndicesQ

prunedPoseSet = poseSet[nonOutlierIndicesAll]

Function removeOutlierIndices(list):
    Do Z-score calculations
    return indices of non-outlier elements in list
Function computeRotationalDistance(q1, q2):
    return angular distance between q1 and q2
```