Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering

Part localization for robotic manipulation

César Sinchiguano

Supervisor: Dr Gaël Écorchard.
May 2019

Acknowledgements

I would like to express my sincere gratitude to .....

Declaration

I hereby declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with methodical instructions for observing the ethical principles in the preparation of university theses. Prague, . May 2019

Abstract

The new generation of the collaborative robots allows the use of small robot arms working with human workers, e.g. the YuMi robot, a dual 7-DOF robot arms designed for precise manipulation of small objects. For the further acceptance of such a robot in the industry, some methods and sensors systems have to be developed to allow them to perform a task such as grasping a specific object. If the robot wants to grasp an object, it has to localize the object relative to itself. This is a task of object recognition in computer vision, the art of localizing predefined objects in image sensor data. This master thesis presents a pipeline for object recognition of a single isolated model in point cloud. The system uses point cloud data rendered from a 3D CAD model and describes its characteristics using local feature descriptors. These are then matched with the descriptors of the point cloud data from the scene to find the 6-DoF pose of the model in the robot coordinate frame. This initial pose estimation is then refined by a registration method such as ICP. A robot-camera calibration is performed also. The contributions of this thesis are as follows: The system uses FPFH (Fast Point Feature Histogram) for describing the local region and a hypothesize-and-test paradigm, e.g. RANSAC in the matching process. In contrast to several approaches those whose rely on Point Pair Features as feature descriptors and a geometry hashing, e.g. voting-scheme as the matching process.

Keywords: Object Detection, Pose Estimation, Robotics, Point Cloud Data

Supervisor: Dr Gaël Écorchard.
Czech Institute of Informatics, Robotics, and Cybernetics, Office
B-323,Jugoslávských partyzánů 3, 160 00 Prague 6

Abstrakt

Nová generace spolupracujících robotů umožňuje použití malých robotických ramen pracujících s lidskými pracovníky, např. robota YuMi, dvojitá robotická ramena 7-DOF určená pro přesnou manipulaci s malými předměty. Pro další přijetí takového robota v průmyslu musí být vyvinuty některé metody a systémy senzorů, které jim umožní provádět úkol, například uchopení určitého objektu. Pokud chce robot uchopit objekt, musí objekt umístit relativně vůči sobě. To je úkol rozpoznávání objektů v počítačovém vidění, což je umění lokalizace předdefinovaných objektů v datech obrazového snímače. Tato diplomová práce představuje potrubí pro rozpoznávání objektů jednoho izolovaného modelu v bodovém mračnu. Systém využívá data z bodového mračna vykreslená z 3D CAD modelu a popisuje jeho charakteristiky pomocí lokálních deskriptorů funkcí. Ty jsou pak porovnány s deskriptory dat z bodového mračna ze scény, aby se 6-DoF pozice modelu v souřadném rámci robota. Tento počáteční odhad pozice je pak vylepšen metodou registrace, jako je ICP. Provádí se také kalibrace robotické kamery. Příspěvky této práce jsou následující: Systém používá FPFH (Fast Point Feature Histogram) pro popis lokální oblasti a hypotézu - a paradigma testu, např. RANSAC v procesu párování. Na rozdíl od několika přístupů k těm, které se spoléhají na vlastnosti Point Pair jako deskriptory vlastností a geometrické hašování, např. hlasovací systém jako proces shody.

Klíčová slova: Detekce objektů, Odhad Pozice, Robotika, Bodová Data

Contents

Figures

Tables

Chapter 1

Introduction

Within this chapter, the reader receives an outline of the general context which surrounds this thesis. Starting with the motivation section and the ultimate goal to be accomplished, and a summary of the thesis' structure follows.

1.1 Motivation

For years, the industrial robot has undergone enormous development. Robot nowadays not only receives a command from the computer. But also has the ability to make decisions itself. Such abilities are well known in the world of the computer vision as recognizing and determining the 6D pose of a rigid body (3D translation and 3D rotation).
However, finding the object of interest or determining its pose in either 2D or 3D scenes is still a challenging task for computer vision. There are many researchers working on it with methods that go from state-of-the-art to deep learning ones where the object is usually represented with a CAD model or object's 3D reconstruction and typical task is the detection of this particular object in the scene captured with RBGD or depth camera. Detection considers determining the location of the object in the input image. This is typical in robotics and machine vision applications where the robot usually does a task like pick-and-place objects. However, localization and pose estimation is a much more challenging task due to the high dimensionality of the search in the workspace. In addition, the object of interest is usually sought in cluttered scenes under occlusion with the requirement of real-time performance which makes the entire task much more difficult.

1.2 Goal

We attempt to provide a system or pipeline for pose estimation of a rigid object in point cloud design for random picking of an isolated object by using depth images acquired from an RGB-D sensor. In addition, the development of a system that can help with the extrinsic calibration of a camera-robot

The goal is just to develop a suitable pipeline for localizing an isolated object where it can be suitable for future work such as a bin-picking system which is out of the scope for this master thesis.

1.3 Thesis structure

The thesis consists of 7 chapters, References and Appendix. The current chapter 1 briefly describes the motivation and the goal of thesis called "Part localization for Robotic Manipulation" which for convenience we refer as 6D pose estimation of a rigid body or pipeline pose estimation interchangeably. Chapter 2 gives a brief introduction to related work, Chapter 3 gives a theoretical background to camera calibration and a gentle description to the main tools used in this thesis such as openCV, open3D, ROS, and software where the CAD model is rendered. Chapter 4 presents the theory as well as every individual step in details of the Robot-camera calibration. Chapter 5 presents the theory as well as every individual step in details of the implemented system, and chapter 6 describes the evaluation of the system. Chapter 7 concludes the thesis and showcases possible future works.

Chapter 2

Related work

Most of the literature tackle the problem of 3D Object Recognition(object detection and 3D pose estimation) by dividing into two broad categories as follow:

1. Global Feature-Based Methods
2. Local Feature-Based Methods

The global feature base methods process the object as a whole for recognition. They define a set of global features which describe the entire 3D object. On the other hand, the local feature based methods extract only local surfaces around specific keypoints. They can handle occlusion and clutter bettere when compared to the global feature-based methods.

2.1 Global Feature-Based Methods

The global feature-based methods define a set of global features which effectively and concisely describe the entire model. Examples of the global feature approach include shape distribution [5], and viewpoint feature histogram [4]. The global feature method ignores all details when it comes to the shape of the object and requires a priori segmentation of the object from the scene. Therefore, they are not suitable for recognition of a partially visible object from cluttered scenes.

2.2 Local Feature-Based Methods

The second class of method, the local feature based methods extract only local surfaces around specific keypoint. Yulan Guo et al. [29] presents a survey of local feature descriptors and cluster them into the three main groups which follow:

1. signature-based,
2. histogram-based, and
3. transform-based methods.

Yulan Guo et al. [29] in his survey claims that local features are much better than global features 2.1 for object recognition in occlusion and clutter scenes. This type of features has also proven to perform better in the area of 2D object recognition. That is why it has been extended to the area of 3D object recognition. Most articles such as [30] and [25] follow this pipeline and compare this with other local descriptors.

Chapter 3

Background

This chapter presents a briefly theoretical background as to mathematical tools and basics of computer vision. In addition, the API and tools used in this thesis. To dive deeply in any topic described ahead, a reference is given.

3.1 Mathematical Tools

3.1.1 Rigid Transformations

A rigid transformation also called Euclidean transformation is a geometric transformation of a Euclidean space that preserves the Euclidean distance between every pair of points. The rigid transformations include rotations, translations, reflections, or their combination. It can be shown that all rigid transformations can be expressed as follows.

$$g_(v) = R \cdot v + t, R \in \mathbb{R}^3 \tag{3.1}$$

A rigid transformation can be represented by using $4x4$ matrices by employing a homogenous coordinates as follows:

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P \\ 1 \end{pmatrix} = \begin{pmatrix} RP + t \\ 1 \end{pmatrix}$$

In the equation 3.1 the matrix, R, is referred to as a rotation matrix and has the following special properties.

- $R = (a\, b\, c)$, $a, b, c \in \mathbb{R}^3$
- $\|a\| = \|b\| = \|c\| = 1$ All columns are unit length
- $a \cdot b = b \cdot c = c \cdot a = 0$ The columns are mutually orthogonal

3.1.2 Rotation Matrices

The matrix R, a set of 3x3 matrices with the following properties, plus the operation of matrix multiplication forms a group called SO(3) which stands for special orthogonal group $\in \mathbb{R}^3$

$R = (a\, b\, c)$, $a, b, c \in \mathbb{R}^3$ is a rotation matrix for $\mathbb{R}^3$ iff

(a) : Astra Camera

(b) : RealSense Camera

Figure 3.1: 2 RGB-D sensors

- $R^T \cdot R = I$
- $det(R) = 1$

Rotation Representations

- A rotation can be expressed as a $3X3$ matrix $R \in SO(3)$ where $R^T \cdot R = R \cdot R^T = I$ and $det(R) = 1$
- A rotation can also be expressed in terms of an angle θ and an axix $\hat{\omega} \in \mathbb{R}^3$ where $\|\hat{\omega}\| = 1$. It can relate to the matrix form via the Rodrigues formula.

$$R = exp(\theta J(\omega)) = I + \sin\theta J(\omega) + (1 - \cos\theta) J(\hat{\omega})^2$$

- And finally a rotation matrix expressed as a unit quaternion:

$$(u_0, u) = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\hat{\omega})$$

3.2 Basics of 3D Computer Vision

3.2.1 RGB-D sensors

Nowadays novel camera systems like the Astra Orbbec and RealSense which provide both color and depth images have become readily available. Therefore, there are great expectations that such sensory devices will lead to a boost of new 3D perception-based applications in the fields of robotics. We are specifically interested in using RGB-D sensors for recognition and localization of an isolated part. In this thesis, both cameras are used. See Figure 3.1 in order to be acquainted with them.

Point Cloud

The receive measurement data from the input sensor get converted in a more generic data structure called point cloud, which is a set of vertices in a three-dimensional coordinate system usually defined by X, Y, and Z coordinates. The vertices are typically intended to represent the external surface of an object. Point clouds can be acquired from hardware sensors such as stereo cameras, 3D scanners, or time-of-flight cameras, or generated from a computer program synthetically. In this thesis, the point cloud is acquired from the sensory devices briefly described above in 3.2.1.

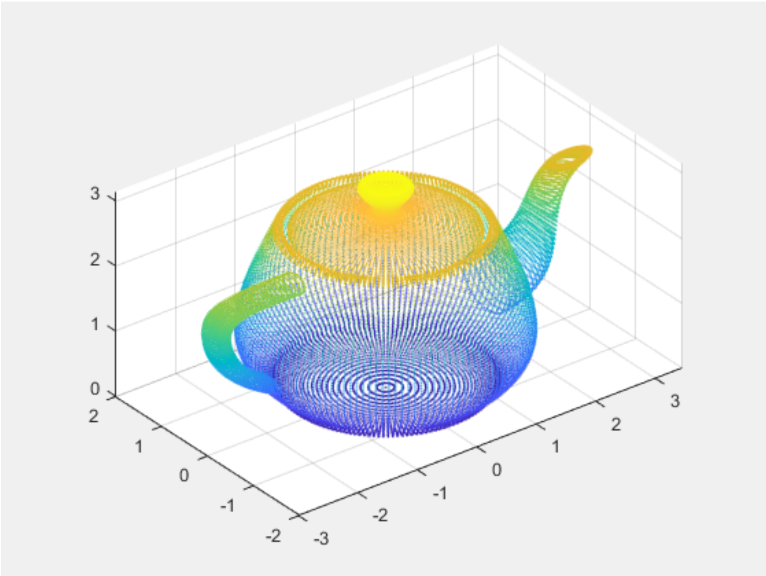


Figure 3.2: Overview of a point cloud (from MathWorks documentation)

3.2.2 Camera Pinhole Model

There are many lens models but Pinhole camera is used in this thesis. A pinhole camera is the simplest model that captures accurately the geometry of perspective projection. The image of the object is formed by the intersection of the light rays with the image plane. An illustration of the pinhole camera is seen in Figure 1. This mapping from the three dimensions onto two dimensions is called perspective projection. The camera projects point in the world frame $P_w = (X, Y, Z)^T \in \mathbf{R}^3$ through the pinhole to the point $p_c = (u, v)$ on the image plane.

3.2.3 Parameters of camera model

We use $(u, v, 1)^T$ to represent a 2D point position in pixel coordinates or image plane. And $(x_w, y_w, z_w, 1)^T$ is used to represent a 3D point position in world coordinates. Note: they were expressed in augmented notation of homogeneous coordinates which is the most common notation in robotics and rigid body transforms. Referring to the pinhole camera model, a camera matrix is used to denote a projective mapping from world coordinates to Pixel coordinates(or image plane), the camera matrix is giving by Eq. 3.2.

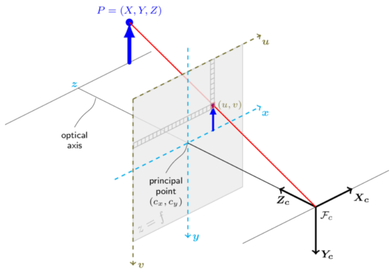


Figure 3.3: View of a Pinhole camera geometry (from Camera Calibration and 3D Reconstruction, openCV)

$$z_c * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K * \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.2}$$

3.2.4 Camera's Intrinsic Parameters

Images coordinates are measured in pixels, normally with the origin in the left upper corner. The focal plane in the pinhole camera model is embedded $\in R^3$ so we need to have a mapping that translates the points in the image plane into pixels, see Figure 3.4.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

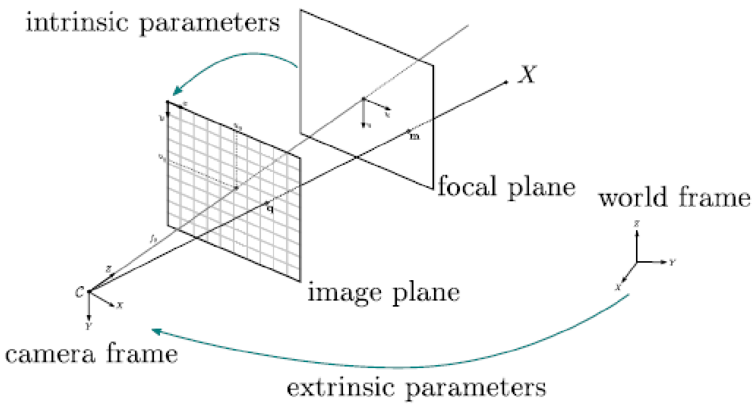


Figure 3.4: Overview of the transformation between the focal plane and the image plane

3.2.5 Camera's Extrinsic Parameters

The transformation between the world coordinate system and the camera coordinate system is achieved be a rotation and a translation. The translation is represented by a vector $t \in \mathbf{R}^3$ and the rotation by a 3x3 orthogonal matrix $\mathbf{R}$. So $\mathbf{R}$ represents a rotation matrix, and it must satisfy the following properties:

$$det(\mathbf{R}) = 1 \tag{3.4}$$

$$\mathbf{R}^T\mathbf{R} = I \tag{3.5}$$

Where I is the identity matrix. The matrix $\mathbf{R}$ and the vector $\mathbf{t}$ altogether are called camera's extrinsic parameters, see Figure.

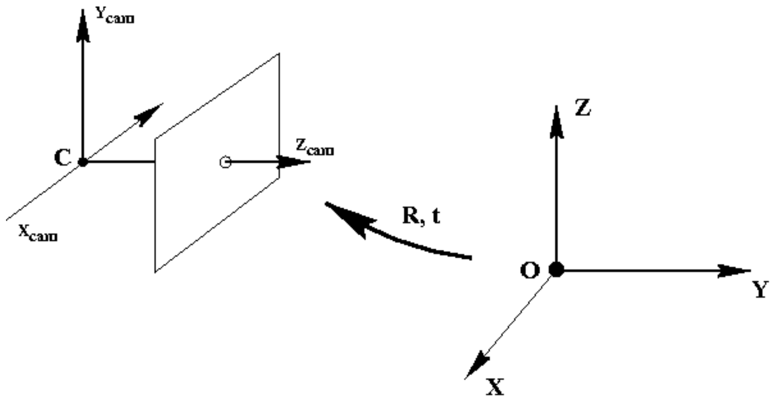


Figure 3.5: Overview of a world coordinate system and camera coordinate system

The transformation of a representation of point in the world coordinate system, $P_w = (X, Y, Z)^T$ into the camera coordinate system, $P_c = (X, Y, Z)^T$ can be done with the following equation.

$$P_c = \mathbf{R}.P_w + \mathbf{t} \tag{3.6}$$

The Equation 3.6 can also be written as:

$$P_c = [\mathbf{R}\ \mathbf{t}] \begin{bmatrix} P_w \\ 1 \end{bmatrix} \tag{3.7}$$

3.3 Robotic Operating System

For this thesis The Robotic Operating System (ROS) is used as main platform. In addition, it is used for visualization purpose and debugging steps. ROS is a flexible framework for writing robot software. In addition, it is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. It is based on the concepts of nodes, topics, messages and services. A node is an executable program that performs computation. Nodes need to communicate with each other to complete the whole task. The communicated data are called messages. ROS provides an easy way for passing messages and establishing communication links between nodes, which are running independently. They pass these messages to each other over a Topic, which is a simple string, Topics are asynchronous communication. As to, a synchronous communication, it is provided by services. Services act in a call-response manner where one node requests that another node execute a one-time computation and provide a response. For more details about ROS, the reader can refer to [6].

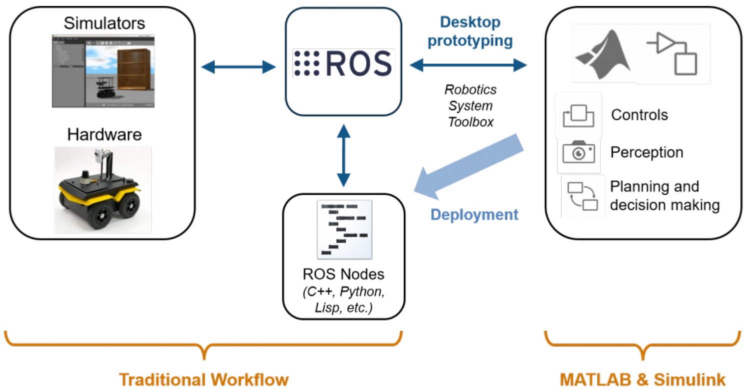


Figure 3.6: A ROS Overview

3.4 Open-source Libraries

3.4.1 PCL

The PCL[7] framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, align 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them.

For different processing steps, a Python bindings for the Point Cloud Library (PCL) is used. This is a reasonable python binding to the point cloud library. At present the following features of PCL, using PointXYZ point clouds, are available;

1. I/O and integration; saving and loading PCD (point cloud data) files
2. segmentation
3. sample consensus model fittting (RANSAC + others, cylinders, planes, common geometry)

4. smoothing (median least squares)
5. filtering (voxel grid downsampling, passthrough, statistical outlier removal)
6. exporting, importing and analysing pointclouds with numpy

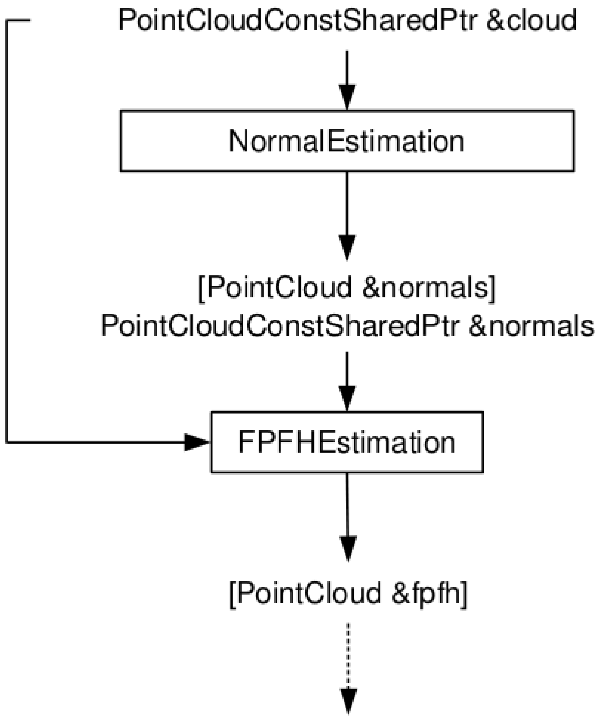


Figure 3.7: An example of the PCL implementation pipeline for Fast Point Feature Histogram (FPFH) [11] estimation.

3.4.2 Open3D

For the purpose of working with any ideal registration algorithm, the Open3D is used in this thesis which is an open-source library that supports rapid development of software that deals with 3D data. The Open3D frontend exposes a set of carefully selected data structures and algorithms in both C++ and Python. Open3D provides data structures for three kinds of representations: point clouds, meshes, and RGB-D images. For each representation, it offers a complete set of basic processing algorithms such as sampling, visualization, and data conversion. In addition, Open3D provides implementations of multiple state-of-the-art surface registration methods, including pairwise global registration, pairwise local refinement as the ICP registration [9], and multiway registration using pose graph optimization.

3.5 Software tools

For the purpose of rendering, convertion and manipulation of any 3D data(CAD model) several tools from the open source communities are used in this thesis such as CloudCompare, MeshLab and FreeCAD.

3.5.1 CloudCompare

CloudCompare is a 3D point cloud (and triangular mesh) processing software. It has been originally designed to perform comparison between two dense 3D points clouds (such as the ones acquired with a laser scanner) or between a point cloud and a triangular mesh. It relies on a specific octree structure dedicated to this task. Afterwards, it has been extended to a more generic point cloud processing software, including many advanced algorithms (registration, resampling, color/normal/scalar fields handling, statistics computation, sensor management, interactive or automatic segmentation, display enhancement, etc.)[12],.

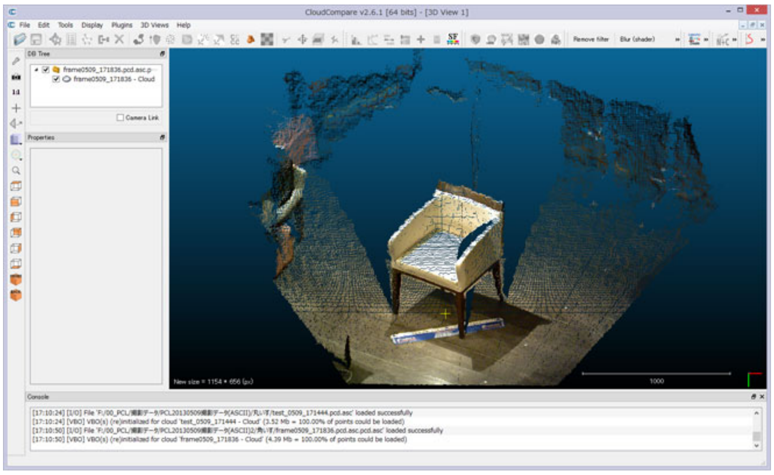


Figure 3.8: CloudCompare (view, edit and process).

3.5.2 MeshLab

Meshlab is an open source system for processing and editing 3D triangular meshes. It provides a set of tools for editing, cleaning, healing, inspecting, rendering, texturing and converting meshes. It offers features for processing raw data produced by 3D digitization tools/devices and for preparing models for 3D printing [13].

3.5.3 FreeCAD

FreeCAD is a 3D CAD/CAE parametric modeling application. It is primarily made for mechanical design, but also serves all other uses where you need to

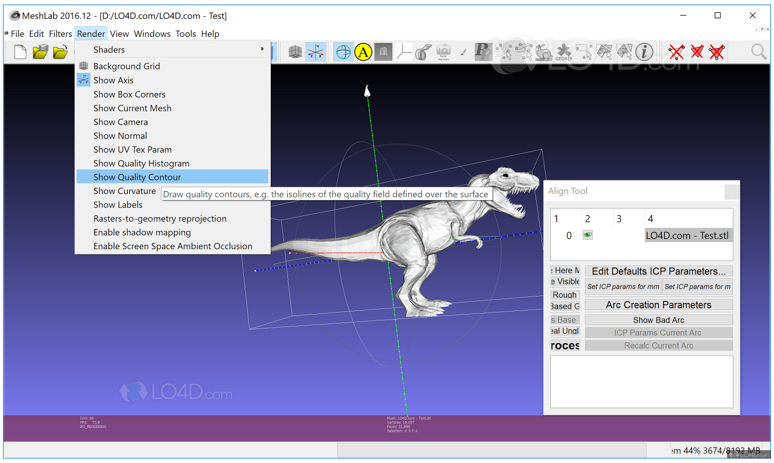


Figure 3.9: MeshLab (view, edit and process).

model 3D objects with precision and control over modeling history [14].

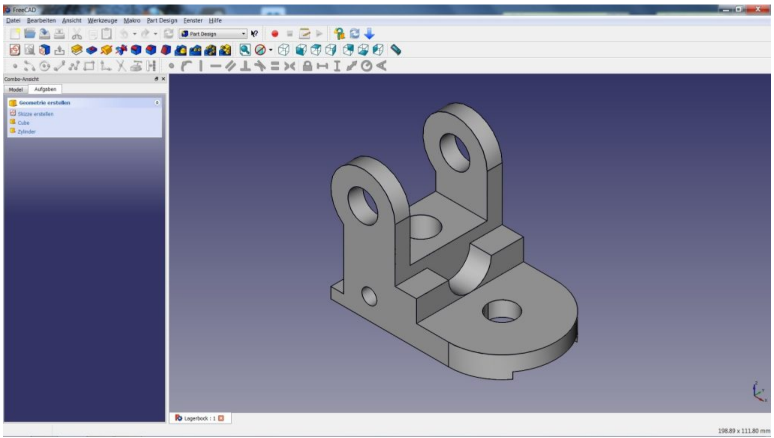


Figure 3.10: A view of the FreeCAD interface.

Chapter 4

Robot-Camera Calibration

This section presents the theory as well as each individual step for estimating the pose of the camera relative to the robot base frame. Such a task is also known as a robot-camera calibration. The robot-camera calibration can be divided into internal camera parameter and external camera parameters also known as Eye-To-Hand calibration. Normally, it is sufficient to perform an internal camera calibration only once. And reliable methods already exist. As to, the Eye-To-Hand calibration is more application specific. It generally requires the position of the camera frame relative to a calibration target frame to be known. Therefore, the proposed method for estimating the robot-camera pose in this thesis is based on tracking a calibration target attached to the end-effector of the robot with known forward kinematics.

4.1 Camera Calibration

Camera calibration is the process of estimating intrinsic and extrinsic parameters. The intrinsic parameters deal with the camera's internal characteristics, such as its focal distance, distortion, and image centre. The extrinsic parameters represent the position and orientation relative to the calibration target. In this thesis the camera calibration is treated separately and can be divided into two main stages:

- Sensor internal parameter calibration such as lens distortion, focal distance, and optical center (image center). In addiction to that, for RGB-D cameras, color and depth image offsets.
- Robot-camera calibration: the pose(position and orientation) of a camera coordinate system in a reference coordinate frame. In this thesis we also refer to it as to Eye-to-Hand calibration. The transformation from the camera coordinate system to the robot base coordinates system (also called world coordinates system interchangeably in this thesis) is shown in Figure4.1

Normally, it is sufficient to perform an internal camera parameter calibration only once for each device unless the lens or sensors itself will be changed or

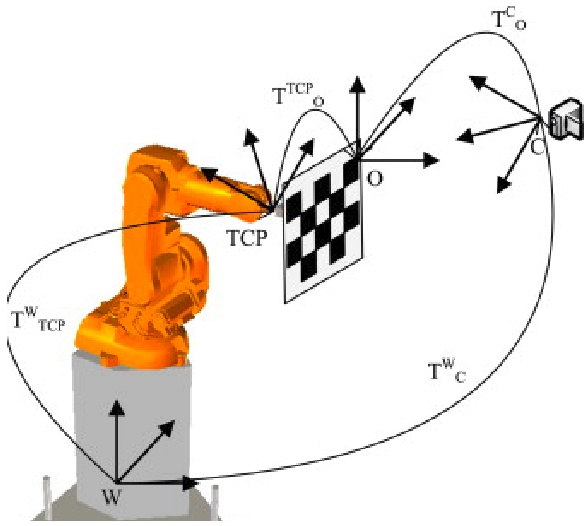


Figure 4.1: Overview of the camera pose estimation system. The system estimates the pose of the camera frame relative to the world frame(also known as robot base frame). Image from [22].

modified. Reliable calibration methods already exist, which are widely used [15] [16].

Robot-camera calibration, on the other hand, is more application specific and an important stage of any 6-DoF pose estimation system according to [23] [24].

4.2 Sensor internal parameter calibration

4.2.1 Camera Model

The choice of camera model influences the final calibration results, so the first step is to select an appropriate camera model. In this thesis, the pinhole camera model 3.2.2 is used. It describes the mathematical relationship between the coordinates of a point in three-dimensional space and its projection onto the image plane of an ideal camera.

The MATLAB, Open CV and the *camera_calibration* ROS [?] packages are the most popular systems for camera calibration. They are already available for checkerboard detection based on the pinhole model and the method proposed by Zhang [15], All of them introduce the radial distortion and tangential distortion. In this thesis, the OpenCV and *camera_calibration* ROS packages are used for the purpose of comparison in this thesis. The technique proposed by Zhang only requires the camera to observe a calibration target shown at a few (at least three) different orientations. the technique relates known points in the world to points in an image, in order to do so, one must first acquire a series of known world points. The most common method is to use known planar objects(checkerboard calibration grid) at different orientations with respect to the camera to develop an independent series of data points. The calibration object chosen in this thesis is a 6x9 checkerboard with the corner points as the known world points as seen in Figure 4.2.

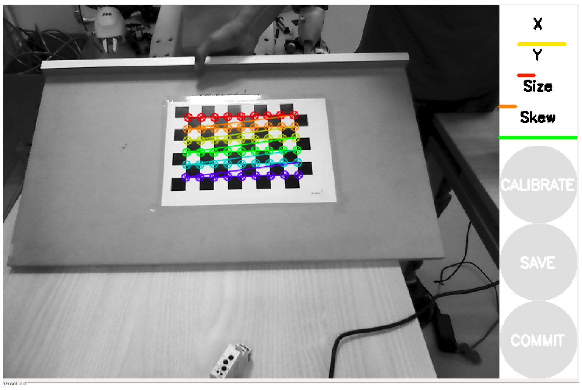


Figure 4.2: Overview of the intrinsic calibration based on industrial calibration ROS package with a 6X9 checkerboard calibration target

4.3 Eye-to-Hand Calibration

In order to know the pose of the camera coordinate system relative to the world coordinate system, also known as robot base frame, extrinsic calibration (estimation of the rotation and translation of the camera frame) method will be used. In this thesis, the method for extrinsic camera calibration based on calibration planer target is used. It is assumed camera intrinsic parameters and distortion coefficients are known in advance 4.2.1 and fixed during the entire sequence. Such a system is shown in Figure 4.3.

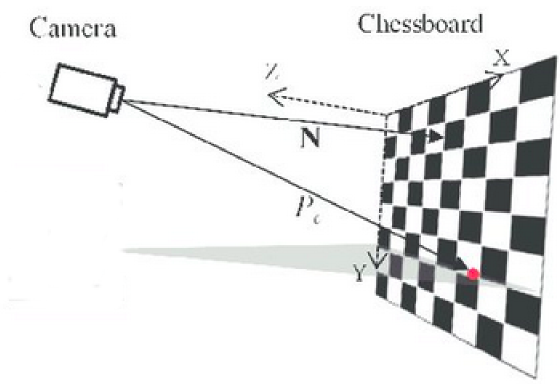


Figure 4.3: Overview of the camera pose estimation system. The system estimates the distance and orientation to the local coordinate system of the checkerboard

4.3.1 Calibration Targets

There are many types of camera calibration targets for use in imaging systems. In this thesis the planar targets are use since they can be easily printed with a standard printer and fixed to a surface. Planar targets can be subdivided as follow:

- Repeated pattern e.g. checkerboard patterns
- Non repeated pattern e.g augmented Reality (AR)

4.3.2 Checkerboard Patterns

Checkerboard calibration targets, where the calibration points are the corner points between squares, are one of the most frequently-used targets. This pattern is simple to produce and allows for high accuracy because the corner points can be detected to subpixel precision. For example, the popular OpenCV library already contains algorithms to automatically locate plain checkerboards. Figure 4.4 shows an example of checkerboard calibration target.

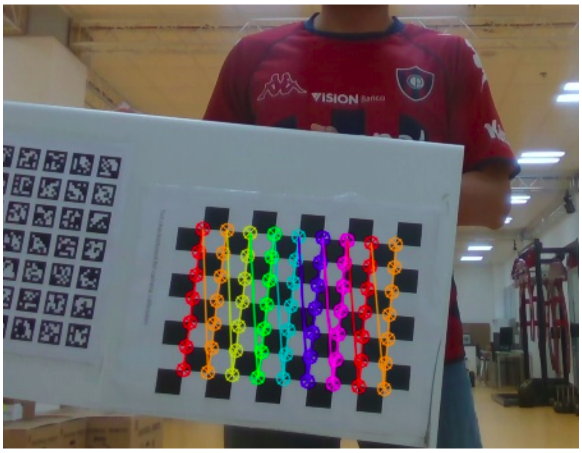


Figure 4.4: Overview of a 7X9 checkerboard calibration grid

4.3.3 Augmented Reality (AR)

AR markers also called Fiducial (individually identifiable) markers have become increasingly popular in recent years. Such markers can be used in a variety of settings such as camera calibration, where small markers are used, those who encode a unique code for identification purposes. There are a large number of markers available. One of the most common fiducial marker designs includes rectangular patterns with identification codes in the interior such as *ARTag*(2005), AprilTag and CALTag to name few. Refer to [20]

Figure 4.5: ARTag, AprilTag and CALTag markers example. Image from [20]

4.3.4 Selection

In this thesis, the checkerboard pattern is used. This pattern is simple to produce and allows for high accuracy because the corner points can be detected to subpixel precision [19]. The calibration target located on the custom-made plate is shown in Figure 4.6

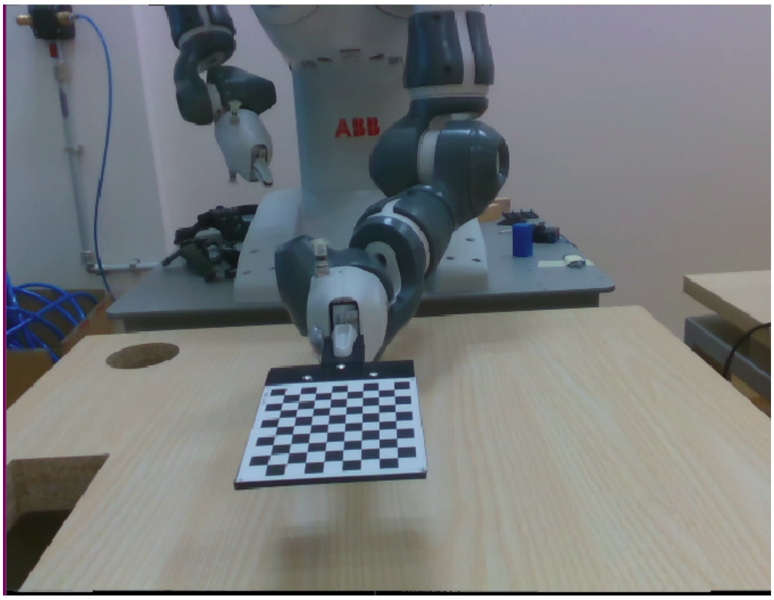


Figure 4.6: An 8X9 Checkerboard Calibration Target fixed on a custom-made plate

4.3.5 Pose Estimation Using A Checkerboard Pattern

The task of estimating the pose of a calibrated camera given a set of **n** 3D points relative to a world and their corresponding 2D projections on the image plane is a fundamental and well-understood topic in computer vision, and it is referred to as a Perspective-n-Point problem in most of the literature. OpenCV provides several methods to solve the Perspective-n-Point problem which returns R (rotation) and t(translation). In order to use the OpenCV capabilities, the image needs to have a suitable format that OpenCV can use. In this thesis, the Robot Operative System is used, which is a suitable platform due to its modular design and rapid integration for a large amount of robot and sensor types. With the help of ROS, the system is split into two nodes also called scripts. The first one which deals with the image acquisition and converting into the right format that OpenCV can use. And the second one, where the algorithm for the pose estimation is implemented. It takes into account the modified image done in the first part. Since a ROS system is modular and each node communicates one another with pass-through messages. The algorithm can be seen in the Algorithm 1.

Data:

1. RGB image data
2. Intrinsic parameters

Result:

1. A $R \in \mathbb{R}^3$ and $t \in \mathbb{R}^3$

$T \leftarrow T_0$;
while $ros :: ok()$ **do**
  if *image* **then**
    Corners are searched in the image scene where a checkerboard is placed with corner detector algorithm already available in OpenCV.;
    The pose of the camera is calculated with the OpenCV algorithm such as solvepnp();
    Publish T, pose, to the ROS network.;
  else
    continue;
  end
end

Algorithm 1: Pose Estimation Using A Checkerboard Pattern

As seen in the 1 the solution to the Perspective-N-Point problem is solved by the OpenCV solvepnp() or solvePnPRansac() functions. Both methods solved the problem by matching a predefined grid of corner locations in the checkerboard to the grid of detected corners in the image plane. These functions need to know the camera matrix and the distortion coefficients in advance. TAs a main feature of the solvePnPRansac function, is that it uses a RANdom SAmple Consensus (RANSAC) method to minimize the error between correspondence points. Both functions can use the following methods to solve the PnP problem:

- $CV\_ITERATIVE(default)$.
- CV_P3P.
- CV_EPNP.

In this thesis, the default one is used. The function outputs a translation and rotation of the object in the camera coordinate system. The rotation is given as 3x1 Rodrigues rotation vector. This later is converted first into a rotation matrix, then to a quaternion which is the standard representation for rotation in ROS. The resulting transformation is published over ROS network for subsequent use. A projection of 3D points expressed in world frame onto the 2D image plane is shown in Figure 4.7. For more details about the solution Perspective-n-point(PnP) refer to [21]

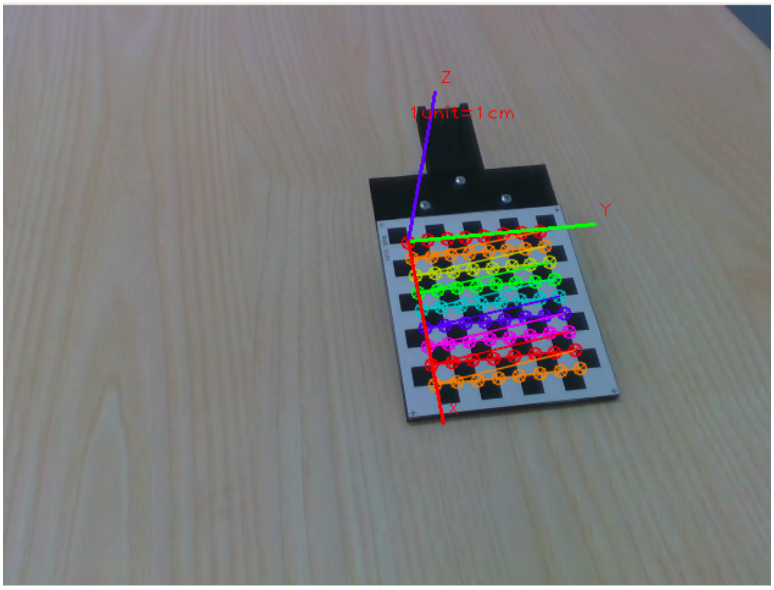


Figure 4.7: Visualization of the 3D world coordinates system projeted onto the 2D image plane

4.3.6 Coordinate Transformation From Robot Base To Camera Frame

From the rigid transformation theory described in 3.1.1, the orientation and translation calculated in 4.3.5, can be represented in a 4x4 matrix by employing homogenous coordinates as follows:

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \tag{4.1}$$

Eq. 4.1 is called the Euclidean transformation, also known as transform. Where $R \in \mathbb{R}^3$ is the rotation matrix and $t \in \mathbb{R}^3$ is the translation vector, altogether representing the pose of the camera frame relative to the calibration target frame.
It is assumed that the transform between the end-effector(or tool centre point) and the robot base, ${}^{R}T_{TCP}$, is known from the forward kinematics of the robot. In addition to that, the transform from the end-effector frame relative

to the calibration target frame, ${}^{TCP}T_T$, was defined according to our need when the custom-made plate, Figure 4.7, was designed.

Since the transform tree is already made, we can retrieve the pose of the camera relative to the robot base frame as follow:

$$ {}^{R}T_C = {}^{R}T_{TCP} \cdot {}^{TCP}T_T \cdot {}^{T}T_C \tag{4.2} $$

In Eq. 4.2, ${}^{R}T_C$ represents the transform from the camera frame relative to the robot base frame. Since the whole system is based on the Robot Operative System (ROS), which is due to its modular design and available integration for a large amount of robot and sensory device, the pose of the camera frame relative the base frame can be also found with the help of tf-ROS package. In Figure 4.8, ${}^{R}T_C$ is shown calculated by software mean.

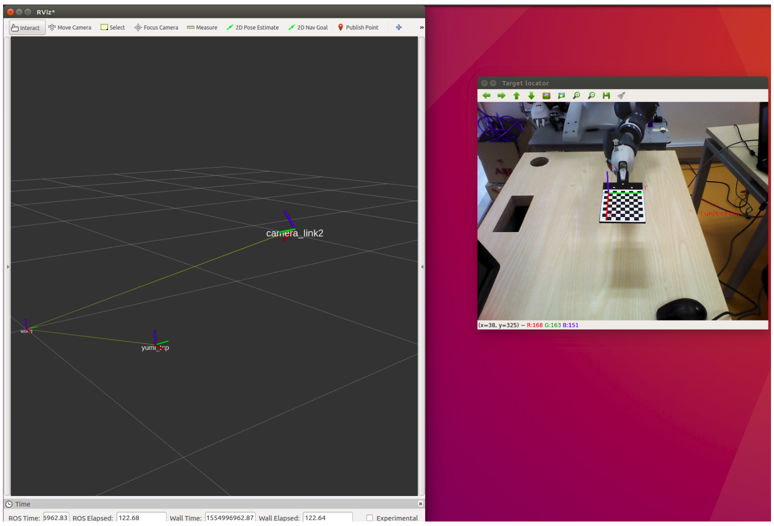


Figure 4.8: Right: Visualization of the transform (camera and target) relative to the robot base frame using ROS Rviz package. Left: Show an image used in the camera pose estimation.

Chapter 5

A 3D Object Pose Estimation Pipeline

This section presents the theory as well as each individual step of the implemented system in detail. The implemented system, or pose estimation pipeline as we refer to interchangeably in this thesis, is fed with two point clouds as input data, one generated from the CAD model and the other one is generated from the output sensory device(RealSense or Astra camera for purpose of comparison).

The 3D CAD model is rendered with the use of software tools described in the previous chapter. The pipeline has two parts, the first part as we refer to as the offline stage where the CAD model is preprocessed, and the second one is an online stage where the point cloud taken from the scene undergoes a preprocessing step similar to the one described above. In addition to, several filter techniques are used in order to segment the ROI (region of interest) and as a final step a matching strategy is applied where it outputs a 6-DOF pose estimation(ground truth) of the object.

5.1 Pose estimation pipeline

Using a local feature base method, the pose estimation pipeline is seen in Figure 5.1. The pipeline used in this thesis is inspired by [1], [2] and [3] with two major modifications. The first one being that the pipeline used in this thesis has the filtering part included in the preprocessing stage in order to better isolate the 3D object. The second modification is related to the matching strategy[25], where in most of the literatures, the preferred strategy is hash table-based voting scheme.A hypothesize-and-test paradigm[25], e.g. RANSAC scheme, is a more suitable method for the purpose of this thesis. For more detail about matching strategies, the reader should refer to the reference.

For the offline stage the 3D CAD model is rendered and it is converted to a point cloud data(PCD) format for a better subsequent use. As to the online stage, the pose estimation pipeline takes as input both clouds, the first one, a cloud from the 3D CAD model and the second one, a cloud from the scene, both clouds are filtered in order to remove noise and outliers. Since this is a tabletop application, we need to extract the candidate point cloud from the table. The table can be removed from the cloud with RANSAC as it was done in [3], which is used to find the largest plane in the image. After, both clouds are downsampled by a Voxel Grid (VG) filtering method, and key points with their features descriptors are computed. Then the two clouds are fed to the matching algorithm where a coerse 6-DoF pose (3 for translation and 3 for rotation) is obtained. The coerse pose is refined with an ICP algorithm registration. Each step is carefully explained in details ahead.

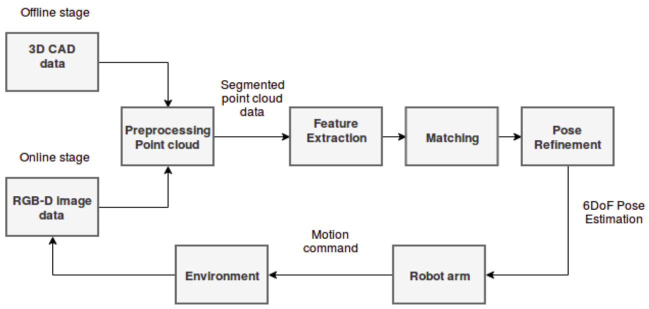


Figure 5.1: General architecture of proposed pose estimation pipeline

5.1.1 Preprocessing stage

After the filtering step, which is only applied to the point cloud representing the scene. The two clouds are downsampled with the technique already implemented in the Point Cloud Library, such as Voxelgrid (VG) filter or Approximate Voxelgrid filtering. This step is required for speeding up the computation process. Since it is a computer vision problem known as Tabletop, it has a dominant plane. Therefore, RANSAC is used to find this dominant plane in the image.

5.1.2 Filtering a Point cloud

The point cloud from the output sensory device contains undesired points, those are often noisy and contains outliers that lead to a high computation time and possibly produces a wrong pose estimation of object. Therefore, it is crucial to remove the noise and outliers from the point cloud in order to obtain accurate point clouds that are suitable for further processing.

In order to identify these suitable point clouds, algorithms for filtering them are already implemented in the Point Cloud Library such as Conditional Removal, Radius/Statistical Outlier Removal, Color Filtering and Passthrough. Since there are few widely used techniques already developed for point cloud filtering. Some of them are aimed at reducing the amount of points in order to speed up the computation time. Others are used to discard outliars. In this thesis we exploit a simple and commonly used filtration pipeline which it has been proven to be an effective combination of methods in several works [26].

Filtering a PointCloud using a PassThrough filter

PassThrough passes points in a cloud based on constraints or threshold for one particular field (X,Y,Z) of the point type. Namely, it removes points where values of selected field are out of range. The filtration pipeline can be seen in the Figure 5.2. For more details and working examples the reader you should refer to [7]

Figure 5.2: Flow Chart of Point Cloud Processing For Filtering Outliars

Plane Segmentation

Since the point cloud from the scene contains undesirable points such as points that represents a table where the object is kept. Further filtering is needed, such filtering is known as plane segmentation. And it is achievable with the RAMSAC based plane fitting method. RAMSAC method finds the largest set of points that fit a plane. The plane equation in three-dimensional point cloud can be defined as:

$$ax + by + cz + d = 0 \tag{5.1}$$

Where the a,b, and c, are the parameters of the plane and d is the distance of the plane from the origin.

RAMSAC selects randomly three points from the dataset and computes the parameters of the corresponding plane, after that it tries to enlarge the plane according to a given threshold,[28]. The step of segmenting the plane in order to remove it is a required condition for the subsequent use. Where a global registration is applied. The method is seen in Algorithm 2 and the flow chart is seen in Figure 5.6

Result: o (object candidate point cloud)
Data: p (3D point cloud), τ , MaxIter, IR
while $t < MaxIter$ - $InlierRatio > IR$ **do**
  Pick 3 points (A, B, C) at random from p ;
  Fit a plane (ax + by + cz + d = 0) to these 3 points;
  AB = B - A;
  AC = C - A;
  N = AB x AC;
  N has the values of (a, b, c);
  Find outlier points o (object candidate points)
  Here, f (x) is plugging in point x into the plane equation divided by the norm of N to measure residual and τ is the threshold
  Find Inlier Ratio as ratio of number of inlier points to total number of points
end

Algorithm 2: RANSAC for plane segmentation [3]

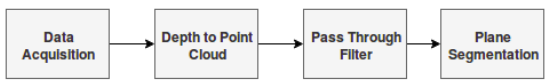


Figure 5.3: Flow Chart of Point Cloud Processing For Plane Segmentation

5.1.3 Extract geometric feature

3D keypoint Detection

Keypoints are relevant points that maintain as much as possible the shape of the object. In order to identify these relevant points, detection methods are used. In addition to that, keypoints are found by sampling the point cloud or downsampling the cloud with VoxelGrid filter.

Voxel Grid filtering method [26] creates a 3D Voxel Grid (3D boxes in 3D space) for each one of the point cloud (model and scene cloud). Then, in each voxel, a point is chosen to approximate all the points that lie on that voxel. Usually, the centroid (an arithmetic mean) of these points or the center (the point in the interior that is equidistant from all points) of this voxel is used as the approximation. The centroid is slower than the center approximation. As a remark, the voxel grid method often drives to geometric information loss. See Figure 5.4 to see the result of applying voxel grid. For more information, the reader should see [27]

Local surface feature description

Vertex normal estimation

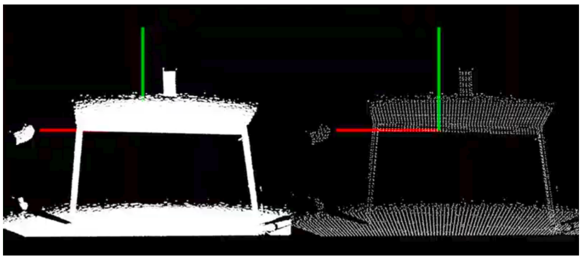


Figure 5.4: Left to Right: Input Image, Output after voxelGrid

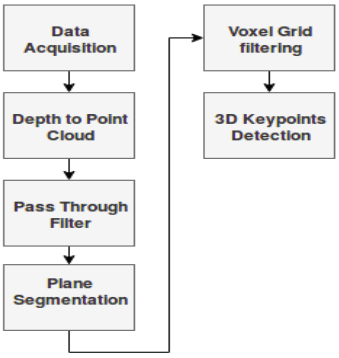


Figure 5.5: Flow Chart of Point Cloud Processing For Keypoints detection

For the subsequent use, normal estimation of both point clouds, model and scene are computed. The approach implemented for computed the normal in this thesis is the vertex normal estimation, a directional vector associated with a vertex, intended as a replacement to the true geometric normal of the surface. The algorithm is already implemented in the open3D [8] library. It computes the normal for every point by finding adjacent points and calculating the principal axis of the adjacent points.

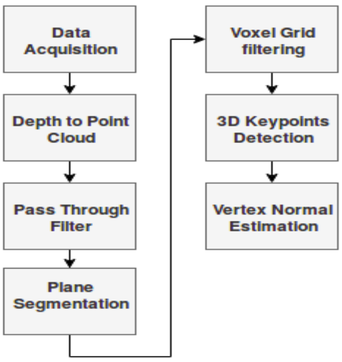


Figure 5.6: Flow Chart of Point Cloud Processing For Plane Segmentation

Local surface feature description

Once the keypoints have been detected for the model and scene point clouds, geometric information of the local surface around those keypoints are extracted and encoded into feature descriptors. According to the approaches employed to construct the feature descriptors in [29], it is classified into three group: signature based, histogram based and transform based method. In this thesis the approach of histogram based method is used. Namely, Fast Point Feature Histogram, FPFH, this method describe the local neighborhood of a keypoint by generating histograms according to the geometric attributes(e.g.,normals) of the local surface.

Figure 5.7: Flow Chart of Point Cloud Processing For Feature Detection

Fast Point Feature Histogram
Fast Point Feature Histogram (FPFH) in [30], is a developed version of the Point Feature Histogram (PFH) [30] with a reduced computational complexity and the same discriminative power.

The generation of a FPFH descriptor consists of two steps. In the first step, a Darboux frame is defined $(u = n_i, v = (p_j \times p_i) \cdot u, w = u \times v)$ for each point pair (p_i and p_j). Then for each query point p it computes only the relationships between itself and its neighbors as follows:

$$\begin{aligned} \alpha &= v \cdot n_j \\ \phi &= \frac{u \cdot (p_j - p_i)}{|p_j - p_i|} \\ \theta &= \arctan(w \cdot n_j, u \cdot n_i) \end{aligned} \tag{5.2}$$

The computation above is called a Simplified Point Feature Histogram (SPFH) which is binned by three angular variations (α, ϕ, θ). Then in the second step, the FPFH of each point is computed using both the SPFH of itself and the weighted ones of its neighbours as follow:

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{n=1}^{k} \frac{1}{\omega_k} SPFH(p_k) \tag{5.3}$$

Where the weight ω_k represents the distance between query point p and a neighbor point p_k in a given metric space.

5.1.4 Searching Strategies

Once both point clouds (object and scene) have been filtered and their shape described, the next step is to find correspondences between them.

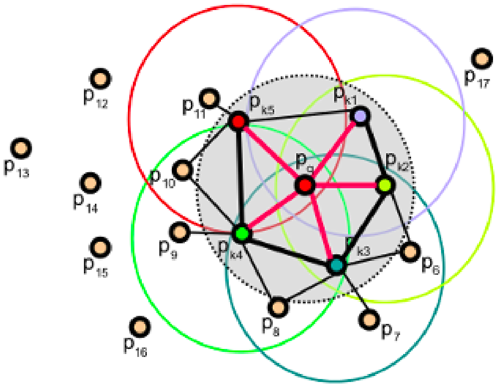


Figure 5.8: The influence region diagram for a Fast Point Feature Histogram. [30]

Therefore, a searching strategy is needed in order to find the proper point corresponding between the two point clouds. Approaches vary in terms of how the correspondence between the scene and model feature is achieved, how a consistent set of matches is derived from the scene-model feature correspondence and how the pose is estimated from a consistent set of correspondence. [25] describes the popular and important approaches to recognition and localization of 3D objects as follow:

1. hypothesize and test
2. matching
3. relational structures
4. Hough(pose) clustering
5. geometry hashing
6. interpretation tree search and
7. iterative model fitting techniques.

In this thesis, a hypothesize-and-test approach is used. In the hypothesize-and-test paradigm, 3D transformation from the object model coordinate frame to the scene coordinate frame is first hypothesize to relate the model feature with the scene features. The transformation is used to verify the match of model features to the scene features. This hypothesized matching is either accepted or rejected depending on the amount of matching error, e.g. RANdom SAmple and Consesus (RANSAC) is a representative method of this approach.

RANSAC-based method

RANdom SAmple and Consesus (RANSAC)[25] is an iterative method designed to find the parameters of a model from a set of data which contains outliers. Given an input noisy data, RANSAC finds the parameters that adjust the input data to a given model, discarding the outliers. In this thesis the RANSAC is used for global registration. In each RANSAC iteration, random points are picked from the model point cloud. Their corresponding points in the scene point cloud are detected by querying the nearest neighbor in the 33-dimensional FPFH feature space. A pruning step takes fast pruning algorithms to quickly reject false matches early. Only matches that pass the pruning step are used to compute a transformation, which is validated on the entire point cloud.

Figure 5.9: Flow Chart of Point Cloud Processing For Finding Correspondence

5.1.5 Local refinement

The last step of the pipeline is the refinement of the alignment achieved by a coarse matching generated in 5.1.4. This step is also commonly referred to as "Fine matching". This alignment is further refined using a surface registration method, such as the Iterative Closest Point (ICP) algorithm, this method is a standard step after the initial estimates for the relative poses due to its robustness and reliability.

Iterative Closest Point

The key concept of the standard ICP algorithm can be summarized as follow:

1. For each point in the source point cloud, finds the closest point in the target point cloud.
2. Estimate the combination of rotation and translation using a root mean square point to point distance metric minimization technique which will best align each source point to its correspondence found in the previous step.
3. Transform the source points using the obtained transformation.
4. Iterate.

Iteratively repeating these steps typically results in convergence to the desired transformation. In most implementations of ICP, the choice of the distance metric which we refer to as d_{max} represents a trade off between convergence and accuracy. A low value results in bad convergence(the algorithm becomes "short sighted"), a large value causes incorrect correspondences to pull the final alignment away from the correct value. The standard ICP algorithm is seen in Algorithm 3. For more details about the ICP and its variants, the reader should refer to [25] and [31]

Standard ICP is seen in Algorithm 3.

Data:

1. Two point clouds: $A = \{a_i\}, B = \{b_i\}$
2. An initial transformation: T_0

Result:

1. The correct transformation, T, which aligns A and B

$T \leftarrow T_0$;
while *not converged* **do**
  for $i \leftarrow 1$ **to** N **do**
    $m_i \leftarrow FindClosestPointInA(T \cdot b_i)$;
    if $\|m_i - T \cdot b_i\| \leqslant d_{max}$ **then**
      $\omega_i \leftarrow 1$;
    else
      $\omega_i \leftarrow 0$;
    end
  end
  $T \leftarrow \underset{T}{\operatorname{argmin}} \sum_{n=1}^{N} \omega_i \|m_i - T \cdot b_i\|^2$;
end

Algorithm 3: Standar ICP

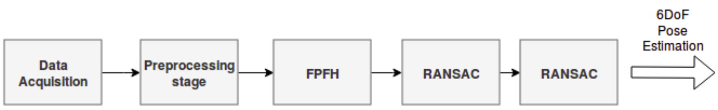


Figure 5.10: Flow Chart of The Pose Estimation Pipeline

Chapter 6
Experimental Results

This chapter presents the experiments and the results when it comes to the evaluation of the following methods. A propose robot-camera calibration method and the 3D object pose estimation method. As to the robot-camera calibration method, the internal parameters of the camera to be calibrated need to be estimated. Methods for estimating the internal parameters also known as intrinsic parameters of the camera already exists and two of the most popular methods available in the open source community were selected. A detail description of those methods is in 4. In order to validate the output of the methods, a reprojection error as the standard metric is selected in this thesis. Then, with the most accurate internal parameters, the camera-robot calibration proceeds. A repeatability test, a validation test for the result of the robot-camera calibration follows. And finally, with the most accurate result of the robot-camera calibration, experiments for testing the 3D pose estimation system follows. For the purpose of testing, an industrial object is required as well as its CAD model. The latter is accomplished with the use of the FreeCAD software 3.5.3, then, a suitable scaling undergoes with the use of CluodCompare software 3.5.1, where a point cloud is generated. As to the validation of the method, a ground truth of the object needs to be known in advance. For such a requirement. A checkboard is used as described in Figure D.1 and Figure D.2. Then, by placing the robot TCP to the specific points (three points in totally), the checkerboard can be localized. With that, a new workspace is produced. By analyzing the translation and rotation errors, the system is evaluated.
The checkboard workspace is used for a rough estimation of the object's pose which is compared with 3D object pose estimation system described in 5.

6.1 Robot-Camera Calibration on the Yumi robot

In order to get the best and accurate estimation of the robot-camera pose, careful attention must be given to the internal parameters of the camera which is a determinant factor of how accurate the extrinsic parameters are. For such estimation of those parameters, two methods were selected and the whole detail to both of them is shown in 4. The Astra camera and the RealSense camera are used in this thesis. The cameras are calibrated with the methods previously mention and their results are shown in ??. As to the validation process, a reprojection error is used. Since it is one of the most used metrics, it is used in this thesis.

Reprojection Error

The reprojection error is the distance between a pattern keypoint detected in a calibration image, and a corresponding world point projected into the same image. Figure 6.4 and Figure 6.2 show the calibration results by analyzing the reprojection error per image when the camera is a RealSense with the OpenCV and camera _industrial calibration methods. As to, Figure 6.3 and Figure 6.1 show the calibration results when the case is with an Astra camera, with the same procedure, by analyzing the reprojection error per image with both methods, the OpenCV and camera _industrial calibration.

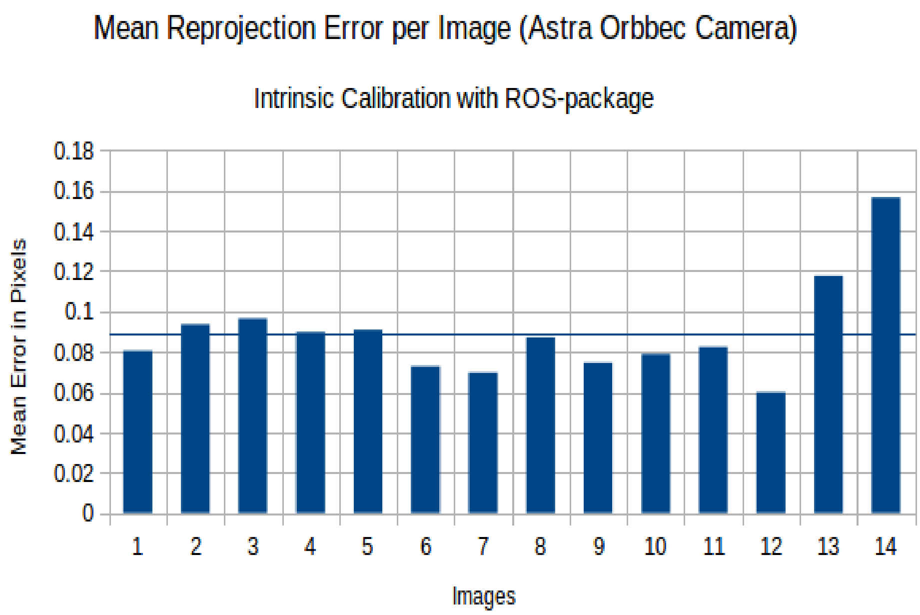


Figure 6.1: Mean Reprojection Error per image with a ROS method (Astra Orbbec Camera)

In order to discuss the result for each case, an average error is calculated for each case. This is done by computing the arithmetical mean of the errors calculated for all the calibration images. And that result should be as close to zero as possible according to the literature in computer vision.

Result Analysis

After computing the average error for both cameras, the results are shown in Table 6.1 related to the Astra camera and Table 6.2 which corresponds to

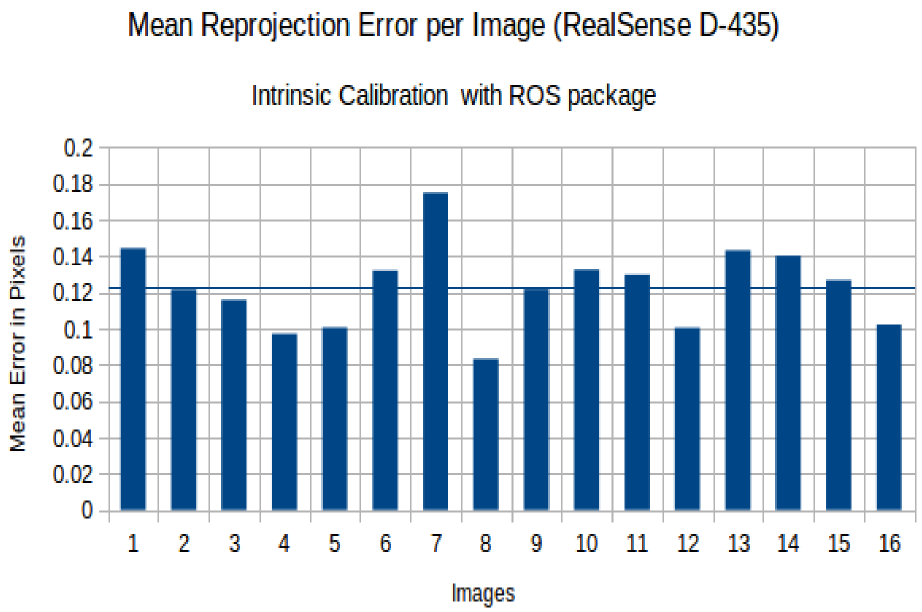


Figure 6.2: Mean Reprojection Error per image with a ROS method (RealSense D-435)

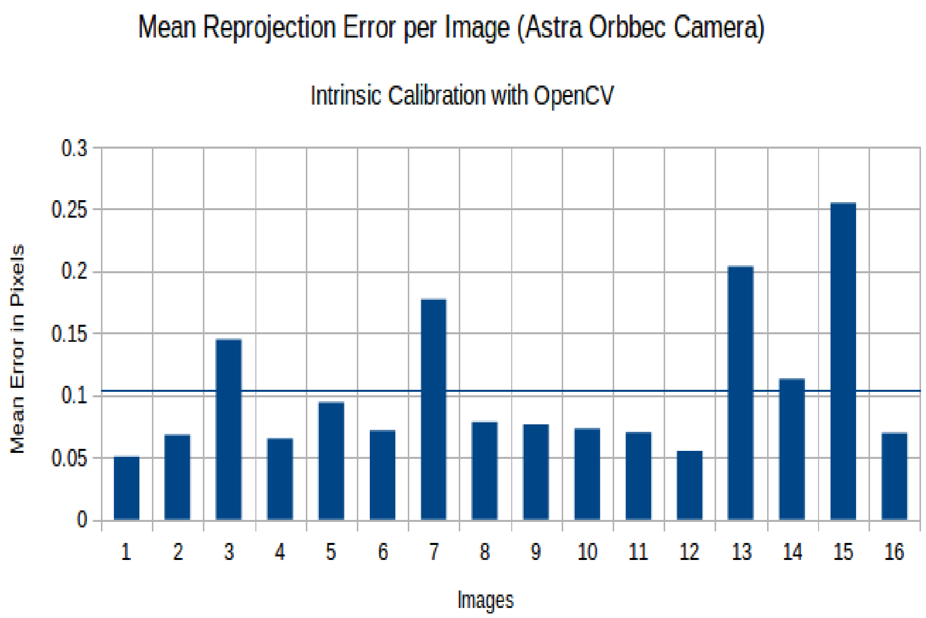


Figure 6.3: Mean Reprojection Error per image with a OpenCV method (Astra Orbbec Camera)

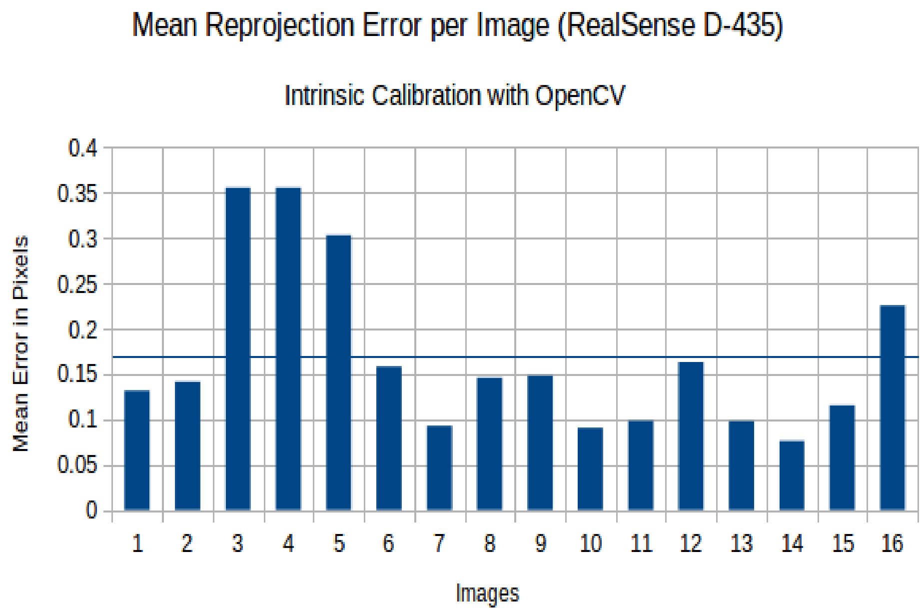


Figure 6.4: Mean Reprojection Error per image with a OpenCV method (RealSense D-435)

the RealSense D-435.
It can be seen that the overall mean error representing the Astra camera is quite correlated to one another. by that we meant, the reprojection error obtained from OpenCV and camera _industrial calibration is very similar with a small offset. But that difference is acceptable due to it holds the requirement to be under the 0.5 pixels, a specified value in the literature for determining how accurate a device sensor is.
On the other hand, the results for the RealSense camera are not quite correlated. Several thoughts can be drawn from that result. The sensor might be quite sensitive to changes in the calibration target resulting in taking wrong measurements. In addition to that, the light condition can also affect the sensory device since the data taken is an RGB image. Such difference presented in the mean reprojection error can hugely affect the good accurate the output of the estimation of the camera frame relative to the robot frame by applying the eye-to-hand calibration. In order to confirm that this different holds. We proceeded to take new measurements, with the careful attention of not moving the calibration target when the sensory device is taken its sample. Then the whole process is evaluated again. The difference holds between methods, but meet the requirements of being under the 0,5 pixels. With the result in hand is quite early to conclude that the RealSense camera is or is not a good choice for solving the main problem of this master Part localization for robotic manipulation. By keeping that difference in mind, we proceed with the next experiment for validating the robot-camera calibration.

As a remark, both cameras were calibrated with same light conditions, with the same calibration target and with an equal number of images. Considering, all in all, it can be concluded that the Astra camera seems to perform well and it can be our final choice for the pose estimation system evaluation.

6.1.1 Eye-To-Hand Calibration

The second experiment in this chapter is related to the eye-to-hand calibration. In this experiment, both cameras, the Astra and RealSense again, are taken into account for the validation test. Since the quality of the extrinsic calibration depends on how good the estimation of the internal parameters is, we proceed with taking the best accurate internal parameters based on the reprojection error as described previously, where the less reprojection error the camera has, the more accurate the camera is.
By defining the best internal parameters, the validation test to run is divided into two types. In the first type of experiment also called a validation test. The calibration plate with a checkerboard place on it as described in **??** is kept at a constant angle, parallel to the XY plane of the robot coordinate system. Figure D.4 shows the basic setup for the extrinsic calibration with a constant orientation parallel to the XY plane of the robot during the whole process that takes it. The whole process includes movements with a small offset between every pose of the TCP (Tool Center Point). Where the joint configuration values for each pose is known in advance. The joint values were saved after each and suitable movement with the help of the handheld of the robot. Each one of those saved movement guarantees that the checkerboard is detected by the robot. Basically, the robot executes a translation of the TCP around the XY plane of itself.
As to the second experiment, where the calibration plate is tilting with predefined orientations. The same approach is taken, several joint values configuration is saved, provided that the checkboard is always detected by the camera to be calibrated. After defining the type of movement, several criteria for obtaining a reasonable estimation of the camera relative to the robot frame are taking into account. One of the consideration is related to the pause in between the movements, is set. This is with the aim to cancel the effects of possible vibrations that the robot can produce so that wrong measurement is avoided during the whole process of the extrinsic calibration. Another consideration is related to speed. A reasonable speed of 25% is set for the validation test.

With the types of experiments knowing in advance, and main criteria to consider for each one of them. The execution of the robot movement follows. To be able to control the robot arm, interface one of both camera at each test, and estimate the pose of the camera relative to the robot base, three nodes were developed as described in 4. The node named as $publishingTF.py$ is responsible for controlling the robot move and publishing the transformation from the robot frame to TCP (Tool Center Point) frame into the ROS network (tf topic to be specific).

The second node named as $target\_locator\_astra.py$ or $target\_locator\_rsense.py$ which depends on the camera used, is responsible for computing the estimation of the camera pose relative to the calibration target (checkerboard) frame. In addition to that, it broadcast that estimation of camera pose into the ROS network. A third node named $listeningTF.py$ is responsible for keeping track of the all coordinate frames over time, and querying for the transformation of the camera frame relative to the robot frame, as well as the computation of the average of the transformation it queries after each executed robot move.

Calibration results

- The following eye-to-hand transform was obtained for the Astra Camera with the calibration plate parallel to the XY plane in robot frame:

$$ {}^{R}T_C = \begin{bmatrix} -2.26051005e-02 & 7.30112611e-01 & -6.82952842e-01 & 1.19260379 \\ 9.99481127e-01 & 8.25583772e-04 & -3.21992981e-02 & 0.12098781 \\ -2.29452788e-02 & -6.83326345e-01 & -7.297752438e-01 & 0.53569317 \end{bmatrix} \tag{6.1} $$

- The following eye-to-hand transform was obtained for the Astra Camera by tilting the calibration plate:

$$ {}^{R}T_C = \begin{bmatrix} -0.01440125 & 0.72431469 & -0.68931911 & 1.19518608 \\ 0.99970886 & -0.00291747 & -0.02395149 & 0.11374275 \\ -0.01935949 & -0.68946336 & -0.7240618 & 0.53606583 \end{bmatrix} \tag{6.2} $$

- The following eye-to-hand transform was obtained for the RealSense D-435 camera with the calibration plate parallel to the XY plane in robot frame:

$$ {}^{R}T_C = \begin{bmatrix} -2.23312402e-02 & 2.94145863e-01 & -9.55499622e-01 & 1.22253213 \\ 9.99723971e-01 & -4.09078692e-04 & -2.34907523e-02 & 0.11776472 \\ -7.30058216e-03 & -9.55760453e-01 & -2.940555535e-01 & 0.32424239 \end{bmatrix} \tag{6.3} $$

- The following eye-to-hand transform was obtained for the RealSense D-435 camera by tilting the calibration plate:

$$ {}^{R}T_C = \begin{bmatrix} -0.018333380.30085851 - 0.95349255 & 1.21972025 \\ 0.99978158 - 0.00405373 - 0.02050249 & 0.09386797 \\ -0.01003355 - 0.95366017 - 0.30071848 & 0.33192816 \end{bmatrix} \tag{6.4} $$

Result Analysis

The propose robot-camera calibration method was successfully performed provided that the calibration target was detected by the 3D camera to be calibrated. In order to validate whether the proposed method is accurate

enough for the pose estimation system described in 4 or not, ground truth is necessary to know in advance. But it was a challenging task to measure an exact orientation and translation of the camera with respect to the robot frame. Given that the housing of the camera (namely the RealSense camera) is rounded which makes it hard to get a good measurement from the mounting point to the camera. In addition to that, the orientation and the translation from the sensor to the housing is not known. It was proved that such difficulty is normal to encounter since the cameras used in this thesis are not suitable for the problem to be solved in this master thesis. Suitable cameras to work with, for the assignment of this thesis are the so-called industrial cameras where the camera frame is given. But that camera was not available. For the given conditions, a validation test is still considered. A rough estimation of the camera pose relative to the robot is measured. The measuring tape was used for the rough estimation but it is not considered as ground truth to evaluate the result of the robot-camera calibration but rather a good idea of what the result should be.
The repeatability test is proposed for the validation of the extrinsic parameters. It consists of repeating over again the whole process when it comes to the estimation of the external values, those who represent the camera pose relative to the robot frame. But a major difference is taking into account for this new test. The difference is that the new joint values are taken and saved provided that the checkerboard is detected by the camera used in turn and the number of movements is increased.
Given the previous condition, the repeatability test is executed. The results are shown in C. Standard deviation and mean values are computed from those results. The mean values and standard deviation for the Astra camera are shown in Table 6.3 and Table 6.4 when the orientation of the camera is kept constant. When it comes to the type of experiments where the robot moves with tilting motion, the results are shown in Table 6.5 and 6.6.
From the values, it can be seen that the external parameters differ from one to another in the range of 1cm for the x-axis and y-axis. As to the z-axis, 1mm is reported. Given that condition, it can be concluded that the external parameters for the Astra camera either with a fast test with few movements or thorough test which means more movements are acceptable. As to the RealSense Camera, where the standard deviation and mean values are shown in Table 6.7 and 6.8 when a constant angle was the case. As to the case where a tilting angle is applied, the results are shown in Table 6.9 and 6.10. From the values, it can be seen that the external parameters differ approximately 1cm for the x-axis, 2cm for the y-axis and 1cm when it comes to vertical displacement. It can be concluded that the Astra is performed constant values in the repeatability test, compare to the RealSense camera. But a new validation test should be applied. By doing intrinsic calibration and the eye-to-hand calibration, the next and final validation test takes place for the 3D pose estimation system.

6.2 Pose Estimation Pipeline

This section presents the experiments and the results and how the validation test was performed for the 3D object pose estimation system. But first of all, for executing such an experiment, few requirements need to be met. The first requirement, a 3D industrial object, namely a textured object is available. As the second requirement, its CAD model is also available since it was created by the author of this thesis with the use of the FreeCAD software 3.5.3. And last but not least, a third requirement, which is related to the pose of the object to be known in advance. This pose should be computed in a different fashion so that the result can be compared with the output of the pose estimation pipeline.
With the output from the pose estimation pipeline and the available pose which it is referred to as ground truth, the accuracy of the system can be determined. But, it was proved to be difficult to determine the real pose of the object for the given thesis research. But a solution needed to comes out for the ground truth refers to as a gold standard pose also in this thesis. Therefore, several methods were discussed knowing that the meaning of ground truth refers to as an information provided by direct observation such as an empirical observation. A simple and unusual solution is taking into account to the ground truth. The use of the checkerboard pattern. Having the checkerboard pattern place on the table, a localization object feature from the robot is used. This is done by position the TCP onto three different points of the checkerboard, by doing so, a new coordinates system is defined where the object is placed to a desired grid of the checkerboard which width and hight is 2cm each. With such setup the object can be move as the author of this thesis wishes with a certain confident of know the deisplacement in XY coordinates of the plane of chessboard. As to the orientation, a digital angle ruler is used in this thesis. The whole setup for the experiment can be seen in Figure D.2 where the checkerboard object is localized onto the top of the table. Figure D.1 shows a rviz simulation, where the relationship between transformations are clearly seen over time.

6.2.1 Validation Test

In this section two type of experiments is taken place. The ultimate goal is to validate the pose estimation pipeline in terms of robustness and accuracy by the means of a distance error and an angle error. The pose estimation pipeline is described in 5, where it said that two 3D datas are required. One of them, a point cloud generated from the CAD model, and the other one, a point cloud generated from the scene, which was adquired from the output of the sensory device used, Astra or RealSense camera. The point cloud generated from the CAD model is called source cloud and the point cloud generated from the depth image is called the target cloud in this thesis.
In order to validate the first experiment, displacements and angles applied to the industrial object are executed. As to displacement, there are three types

related to it, one in the x-axis, the other one in the y-axis and the last one in the XY plane of the chekerboard. The distance of such a displacement is 2cm each. Since the transformation of the checkboard is known in advance plus the displacement applied to the object, the rough object pose can be known by visul inspection of the author of this thesis. When it comes to the validation test for the angle, a digital angle ruler is used, and the orientation is applied on the z-axis, namely a clockwise rotation. For the validation of the second experiment, the same principle is applied as described above when it comes to change the pose of the objec around the XY plane of the checkerboard. But a major difference exists for the source point cloud (CAD model). The CAD model is substitute for a partially view point cloud generated the camera used. The change was applied in order to see whether exist or not an improvement. Since the pose estimation system from time to time failed for the given conditions in the first experiment.

Calibration results

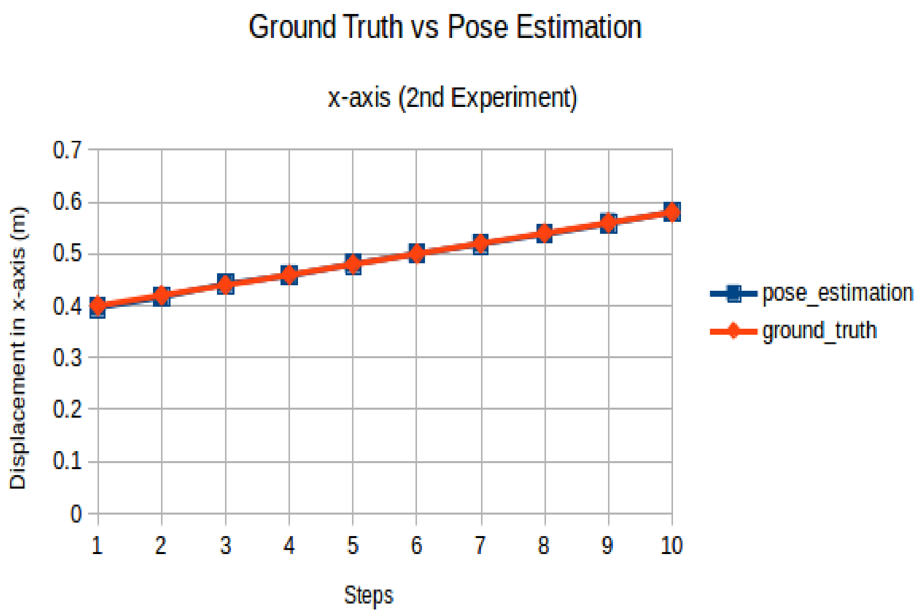


Figure 6.5: Ground Truth vs Pose Estimation: x-axis (2nd Experiment)

Result Analysis

lll

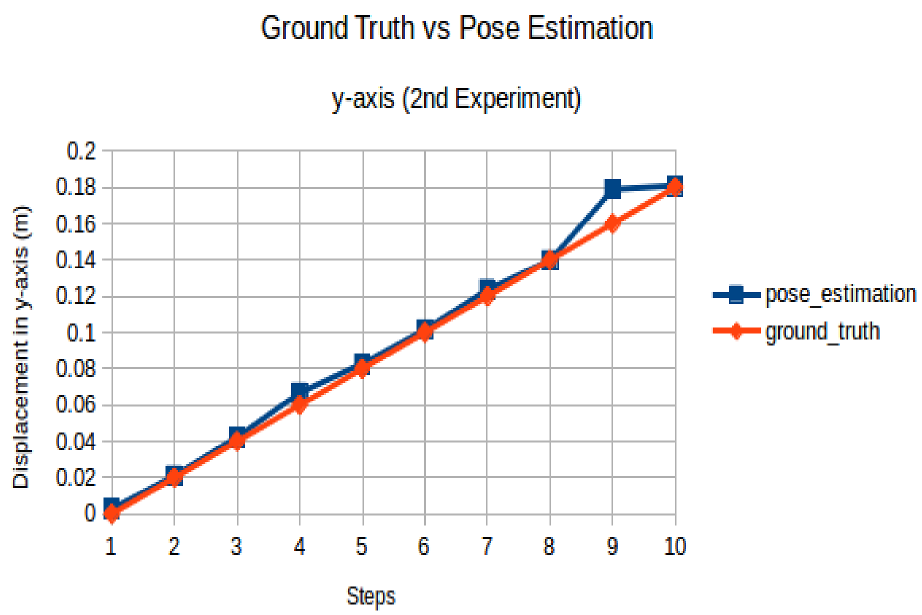


Figure 6.6: Ground Truth vs Pose Estimation: y-axis (2nd Experiment)

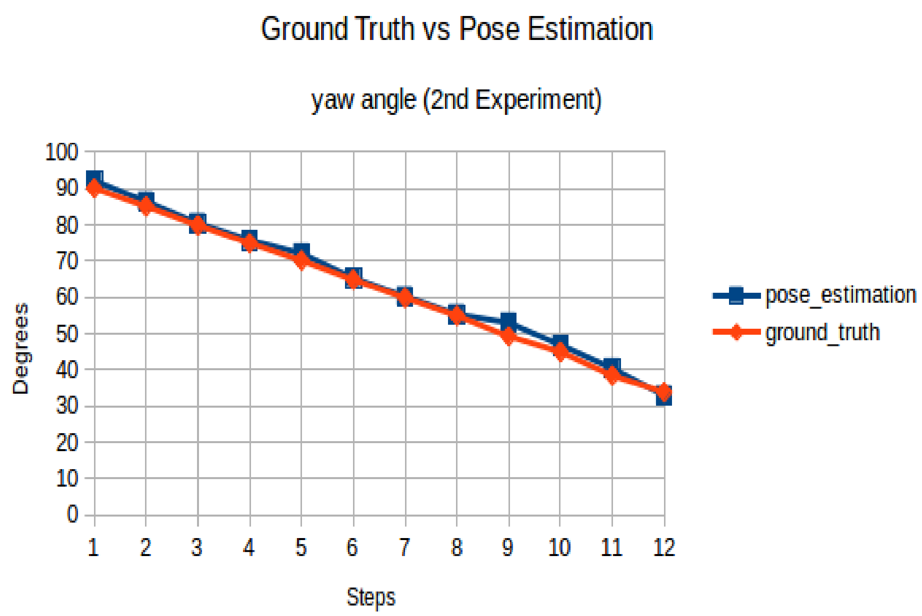


Figure 6.7: Ground Truth vs Pose Estimation: yaw angle (2nd Experiment)

Method	Overall Mean Error
OpenCV	0.1041954808
ROS	0.1081118023

Table 6.1: Experiment data for internal Astra sensor calibration.

Method	Overall Mean Error
OpenCV	0.1684388411
ROS	0.122868849

Table 6.2: Experiment data for internal RealSense sensor calibration.

$x[m]$	$y[m]$	$z[m]$	q_x	q_y	q_z	q_ω
1.1926	0.1245	0.5368	0.653175	0.661978	-0.269761	-0.249755

Table 6.3: Mean Values of The Repeatability Test With a Constant Orientation of the Calibration Plate(Astra Camera).

$\sigma_{x[m]}$	$\sigma_{y[m]}$	$\sigma_{z[m]}$	σ_{q_x}	σ_{q_y}	σ_{q_z}	σ_{q_ω}
0.011012	0.009877	0.000906	0.000243	8.45E-05	0.000446	0.000976

Table 6.4: Standard Deviation from The Repeatability Test With a Constant Orientation of the Calibration Plate(Astra Camera).

$x[m]$	$y[m]$	$z[m]$	q_x	q_y	q_z	q_ω
1.197089	0.116571	0.539509	0.653488	0.660281	-0.270449	-0.252499

Table 6.5: Mean Values of The Repeatability Test With Tilting Motion of the Calibration Plate(Astra Camera).

$\sigma_{x[m]}$	$\sigma_{y[m]}$	$\sigma_{z[m]}$	σ_{q_x}	σ_{q_y}	σ_{q_z}	σ_{q_ω}
0.012046	0.010473	0.008465	0.000837	0.002155	0.000920	0.002450

Table 6.6: Standard Deviation from The Repeatability Test With Tilting Orientation of the Calibration Plate(Astra Camera).

$x[m]$	$y[m]$	$z[m]$	q_x	q_y	q_z	q_ω
1.222425	0.112532	0.324441	0.564006	0.573552	-0.426780	-0.413272

Table 6.7: Mean Values of The Repeatability Test With a Constant Orientation of the Calibration Plate(RealSense Camera).

$\sigma_{x[m]}$	$\sigma_{y[m]}$	$\sigma_{z[m]}$	σ_{q_x}	σ_{q_y}	σ_{q_z}	σ_{q_ω}
0.000160	0.007729	0.000294	9.20E-05	4.19E-05	5.26E-05	1.31E-05

Table 6.8: Standard Deviation from The Repeatability Test With a Constant Orientation of the Calibration Plate(RealSense).

$x[m]$	$y[m]$	$z[m]$	q_x	q_y	q_z	q_ω
1.222425	0.112532	0.324441	0.564006	0.573552	-0.426780	-0.413272

Table 6.9: Mean Values of The Repeatability Test With Tilting Motion of the Calibration Plate(Real Sense).

$\sigma_{x[m]}$	$\sigma_{y[m]}$	$\sigma_{z[m]}$	σ_{q_x}	σ_{q_y}	σ_{q_z}	σ_{q_ω}
0.000160	0.007729	0.000294	9.20E-05	4.19E-05	5.26E-05	1.31E-05

Table 6.10: Standard Deviation from The Repeatability Test With Tilting Orientation of the Calibration Plate(RealSense Camera).

Chapter 7
Future Work

Chapter 8

Conclusions

Bibliography

[1] Cheng-Hei Wu, Sin-Yi Jiang and Kai-Tai Song*, "CAD-Based Pose Estimation for Random Bin-Picking of Multiple Objects Using a RGB-D Camera" in: International Conference on Control, Automation and Systems (ICCAS 2015),Oct. 13-16,2015 in BEXCO, Busan, Korea,https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7364621.

[2] Ajmal S. Mian, Mohammed Bennamoun, and Robyn Owens, "Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes" in: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 28, NO. 10, OCTOBER 2006,http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.6478&rep=rep1&type=pdf.

[3] Nitin J. Sanket and Daniel D. Lee, "6D Object Pose Estimation using RGBD Data and Fast-ICP" , https://nitinjsanket.github.io/project/ese650/p6/nitinsan_project6.pdf.

[4] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, John Hsu, "Fast 3D recognition and pose using the Viewpoint Feature Histogram" in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems , https://ieeexplore.ieee.org/document/5651280.

[5] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin, "Shape Distributions", http://graphics.stanford.edu/courses/cs468-08-fall/pdf/osada.pdf.

[6] Quigley, Morgan., Conley, Ken., Gerkey, Brian P.., Faust, Josh., Foote, Tully., Leibs, “ROS: an open-source Robot Operating System” in: Conference Paper; 2009. http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

[7] Radu Bogdan Rusu and Steve Cousins, “3D is here: Point Cloud Library (PCL)” in: IEEE International Conference on Robotics and Automation (ICRA); 2011. http://pointclouds.org/assets/pdf/pcl_icra2011.pdf

[8] Qian-Yi Zhou and Jaesik Park and Vladlen Koltun, "Open3D: A Modern Library for 3D Data Processing" arXiv:1801.09847; 2018. http://www.open3d.org/

[9] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes"in: PAMI:1801.09847;1992.

[10] Klaas Klasing, Daniel Althoff, Dirk Wollherr and Martin Buss, "Comparison of Surface Normal Estimation Methods for Range Sensing Applications"

[11] Radu Bogdan Rusu, Nico Blodow, Michael Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration" in: IEEE International Conference on Robotics and Automation, Kobe International Conference Center Kobe, Japan, May 12-17, 2009.

[12] CloudCompare, https://www.danielgm.net/cc/

[13] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool" in: Sixth Eurographics Italian Chapter Conference, page 129-136, 2008, http://www.meshlab.net/.

[14] FreeCAD, https://www.freecadweb.org/

[15] Zhengyou Zhang, "A Flexible New Technique for Camera Calibration" in: IEEE Transactions on Pattern Analysis and Machine Intelligence, Nov 2000 , https://ieeexplore.ieee.org/document/888718.

[16] Berthold K.P. Horn, "Tsai's camera calibration method revisited", 2000 ,https://www.researchgate.net/publication/238495425_Tsai%27s_camera_calibration_method_revisited.

[17] camera calibration ROS, http://wiki.ros.org/camera_calibration?distro=melodic.

[18] Y. M. Wang, Y. Li, and J. B. Zheng, "A Camera Calibration Technique Based on OpenCV", https://ieeexplore.ieee.org/document/888718.

[19] B. Atcheson, F. Heide, and W. Heidrich, "CALTag: High Precision Fiducial Markers for CameraCalibration" in: Vision, Modeling, and Visualization, 2010, https://pdfs.semanticscholar.org/2dba/e046717b058382a5a04f800405f92d040200.pdf.

[20] Artur Sagitov, Ksenia Shabalina, Leysan Sabirova, Hongbing Li, and Evgeni Magid, "ARTag, AprilTag and CALTag Fiducial Marker Systems: Comparison in a Presence of Partial Marker Occlusion and Rotation", https://pdfs.semanticscholar.org/4b31/a78ff17b43b27afbddc63e0eede94bcec334.pdf?_ga=2.61993516.127748258.1556187821-1484823862.1553547725.

[21] OpenCV PnP problem, https://docs.opencv.org/3.4.3/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d.

[22] Bernard Schmidt and Lihui Wang, "Automatic work objects calibration via a global–local camera system" in: obotics and Computer-Integrated Manufacturing, 2014, https://www.sciencedirect.com/science/article/pii/S0736584513001014.

[23] J. Ilonen , V. Kyrki , "Robust robot-camera calibration" in: International Conference on Advanced Robotics (ICAR),2011, https://ieeexplore.ieee.org/document/6088553.

[24] Justinas Miseikis,Kyrre Glette,Ole Jakob Elle and Jim Torresen, "Automatic Calibration of a Robot Manipulator and Multi 3D Camera System" in: obotics and Computer-Integrated Manufacturing, 2014, https://www.researchgate.net/publication/289587273_Automatic_Calibration_of_a_Robot_Manipulator_and_Multi_3D_Camera_System.

[25] ANIL K. JAIN and CHITRA DORAI, "3D object recognition: Representation and matching" in: Statistics and Computing(2000)10,167–182, https://epdf.tips/3d-object-recognition-representation-and-matching.html.

[26] Xian-Feng Han *, Jesse S. Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, Liping Xiao, "A review of algorithms for filtering the 3D point cloud" in: Signal Processing Image Communication, May 2017, https://www.freecadweb.org/.

[27] Carlos Moreno and Ming Li, "A Comparative Study of Filtering Methods for Point Clouds in Real-Time Video Streaming" in: Proceedings of the World Congress on Engineering and Computer Science 2016 Vol I, October 19-21, 2016, San Francisco, USA, http://www.iaeng.org/publication/WCECS2016/WCECS2016_pp389-393.pdf.

[28] Rifat Kurban, Florenc Skuka, Hakki Bozpolat, "Plane Segmentation of Kinect Point Clouds using RANSAC" in: CIT 2015 The 7th International Conference on Information Technology, https://www.semanticscholar.org/paper/Plane-Segmentation-of-Kinect-Point-Clouds-using-Kurban-Skuka/5f8d9a32a21db5a7aa5bc68ce8ee8e486002ea06.

[29] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan, "3D Object Recognition in Cluttered Sceneswith Local Surface Features: A Survey" in: IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 36, NO. 11, NOVEMBER 2014, https://www.semanticscholar.org/paper/3D-Object-Recognition-in-Cluttered-Scenes-with-A-Guo-Bennamoun/fa50e835690611b81929714602f06048ca1c2012.

[30] Radu Bogdan Rusu, Nico Blodow, Michael Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration" in: IEEE International Conference on Robotics and Automation, 2009, https://ieeexplore.ieee.org/document/5152473.

[31] Sebastian Thrun,Dirk Haehnel,Aleksandr V. Segal, "Generalized-ICP", http://www.robots.ox.ac.uk/~avsegal/resources/papers/Generalized_ICP.pdf.

Appendix A

List of Notation

Symbol	Meaning
$\mathbb{R}$	The real numbers
ICP	Iterative Closest Point
DOF	Degree(s) of Freedom.
CAD	Computer Aided Design.
FPFH	Fast Point Feature Histogram.
PCL	The Point Cloud Library is an open-source library of algorithms for point cloud processing tasks and 3D geometry processing.
Open3D	Open3D is an open-source library that supports rapid development of software that deals with 3D data.
RGB-D Camera	Specific type of depth sensing device that work in association with a RGB camera.
RANSAC	Random sample consensus. An iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers.
ROS	The Robot Operating System is a set of software libraries and tools that help you build robot applications.
ToF	Time-Of-Flight denotes a variety of methods that measure the time that it takes for an object, particle or wave to travel a distance through space.

Appendix B

Assignment of this thesis

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Sinchiguano Chiriboga Cesar Augusto**	Personal ID number:	**464328**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Part localization for robotic manipulation

Master’s thesis title in Czech:

Lokalizace předmětů pro robotickou manipulaci

Guidelines:

The new generation of so-called collaborative robots allow the use of small robotic arms without them being isolated from human workers. Such an example of collaborative robot is the YuMi robot, a dual 7-axis arms robot designed for precise manipulation of small parts and available in our laboratory.
For further acceptance of such robots in the industry, some methods and sensor systems have to be developed to allow them to pick parts without the position of the parts being known in advance, just as humans do.
The aim of the project is to implement algorithms for the localization of known parts. Part of the work will consist in calibrating the camera relatively to the robot and developing methods to obtain the ground truth position of parts. The second part will consist in developing the localization algorithms themselves.
The student‘s tasks will consist in:
- developing a camera-robot calibration algorithm,
- developing the software and/or hardware to determine the ground-truth position of a single isolated part,
- developing algorithms to localize an isolated part,
- verifying the system experimentally,
- studying the possibility to extend the system to picking multiple isolated part and random bin picking.

Bibliography / sources:

[1] Besl PJ, McKay ND. A method for registration of 3-D shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 1992; Vol 14(2), pp. 239–256.
[2] Point cloud registration from local feature correspondences—Evaluation on challenging datasets
Petricek T, Svoboda T (2017) Point cloud registration from local feature correspondences—Evaluation on challenging datasets. PLOS ONE 12(11): e0187943. https://doi.org/10.1371/journal.pone.0187943
[3] Segal AV, Haehnel D, Thrun S. Generalized-ICP. In: Robotics: Science and Systems V. Seattle, USA; 2009.
[4] Mian AS, Bennamoun M, Owens RA. Automatic Correspondence for 3D Modeling: An Extensive Review. International Journal of Shape Modeling. 2005; Vol. 11(02): pp. 253–291.

Name and workplace of master’s thesis supervisor:

Dr. Gaël Pierre Marie Ecorchard, Intelligent and Mobile Robotics, CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **24.10.2018** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until:
by the end of summer semester 2019/2020

Dr. Gaël Pierre Marie Ecorchard	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt

Student's signature

Appendix C

Eye-To-Hand Calibration Results

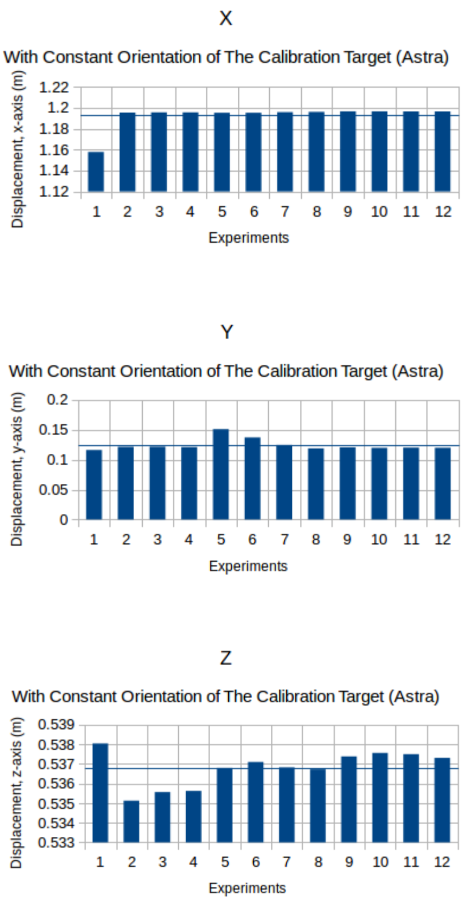


Figure C.1: Eye-To-Hand Result with a Constant Orientation of the Calibration Plate(Astra Camera)

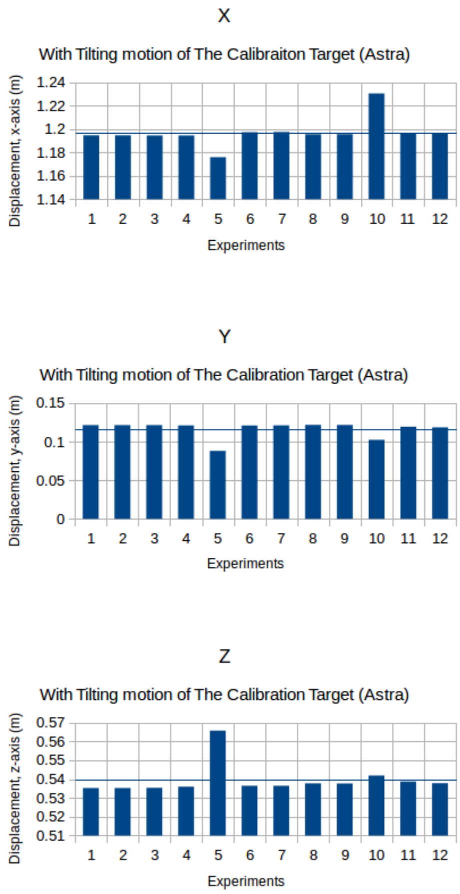


Figure C.2: Eye-To-Hand Result with Tilting Motion of the Calibration Plate(Astra Camera)

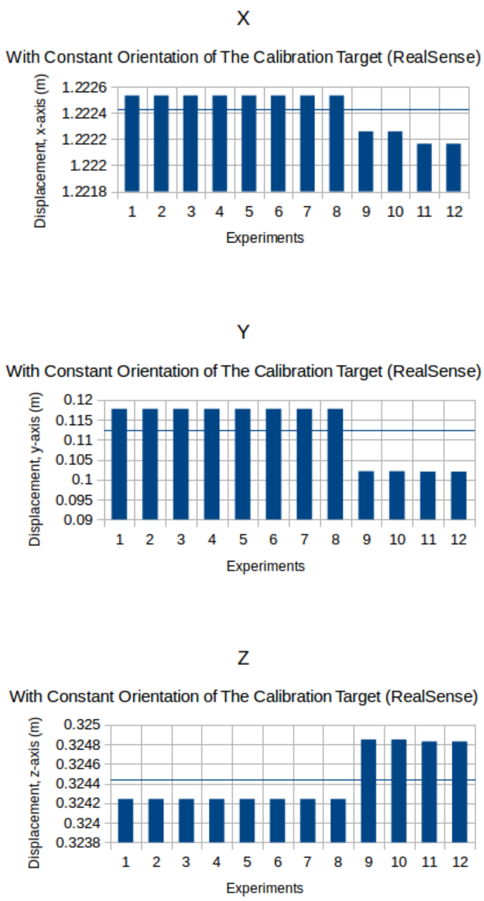
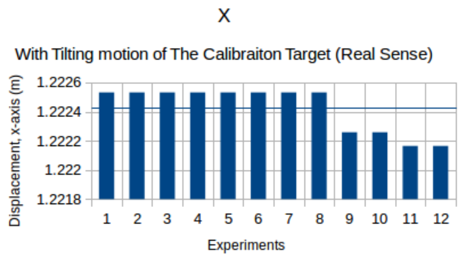
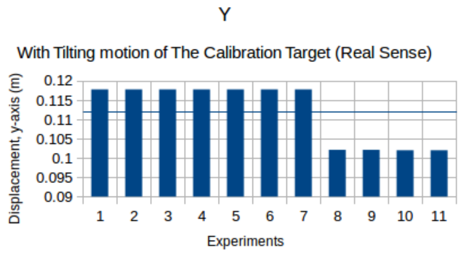


Figure C.3: Eye-To-Hand Result with a Constant Orientation of the Calibration Plate(RealSense)





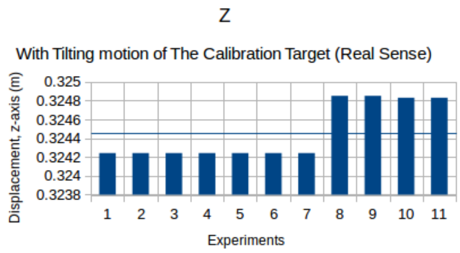


Figure C.4: Eye-To-Hand Result with Tilting Motion of the Calibration Plate(RealSense Camera)

Appendix D

Images

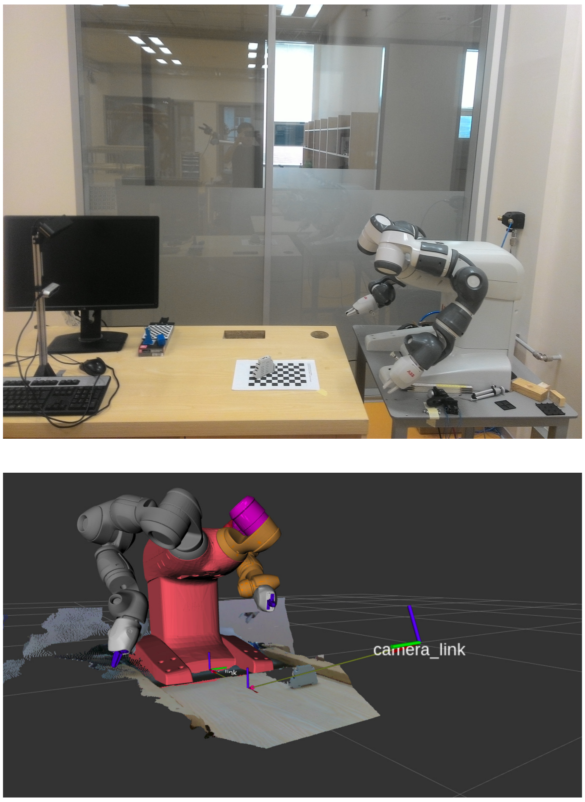


Figure D.1: Overview of the Validation System 1

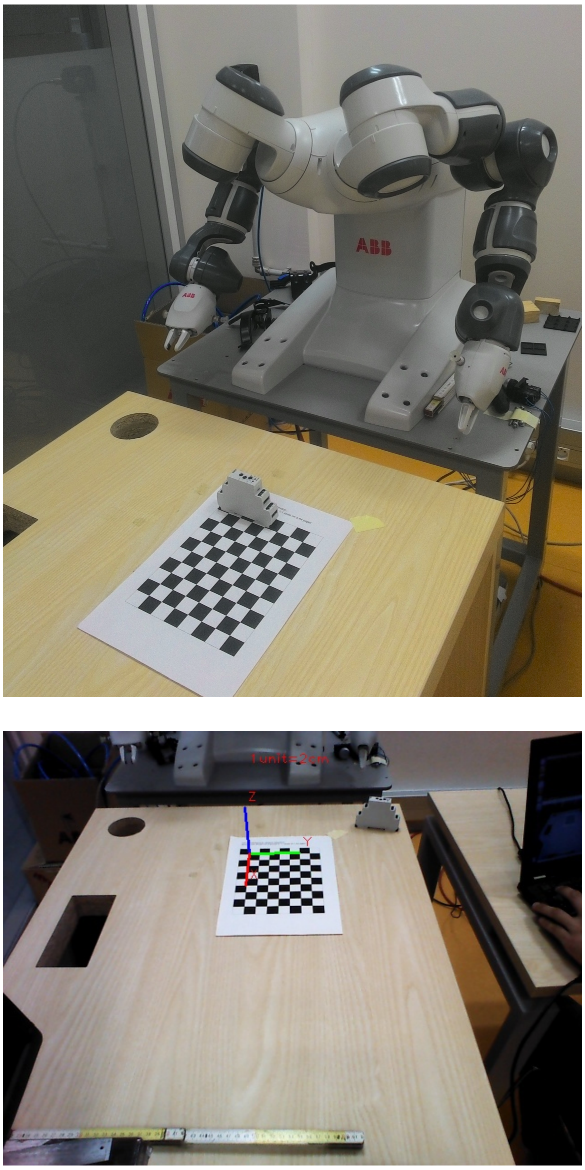


Figure D.2: Overview of the Validation System 2

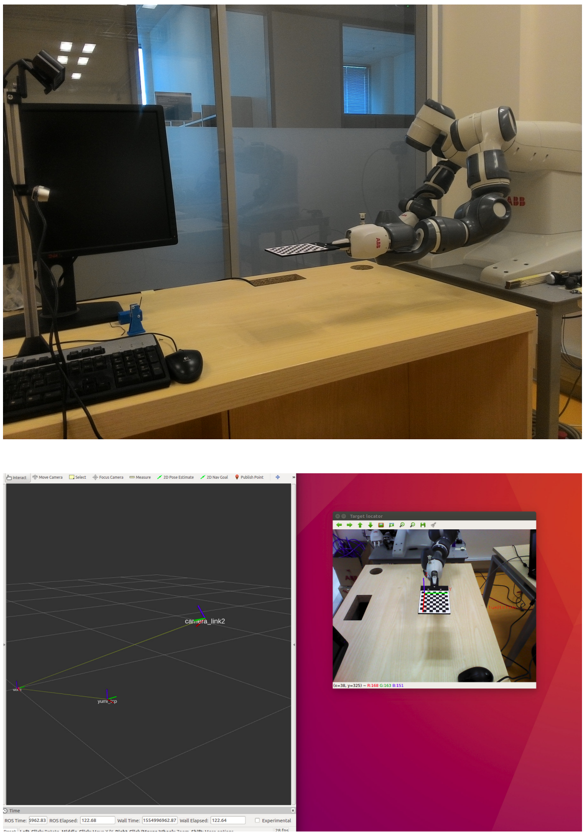


Figure D.3: Setup for the Validation Test with a Constant Orientation

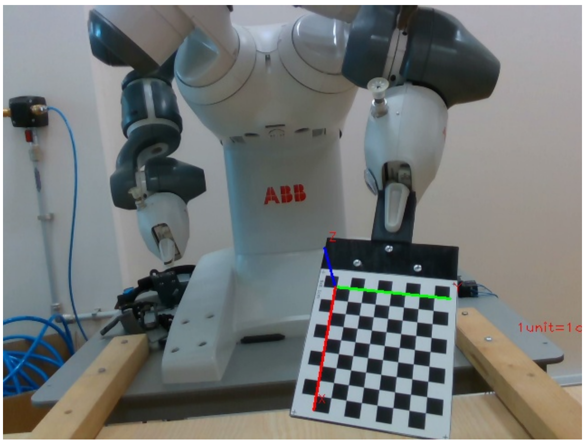


Figure D.4: Setup for the Validation Test with Tilting Motion