Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering

Part localization for robotic manipulation

César Sinchiguano

Supervisor: Dr Gaël Pierre Écorchard.
May 2019

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr Gaël Pierre Écorchard for his patient guidance and good predisposition to help me with my thesis.

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 allow the use of small robot arms working in an asynchronous or synchronous fashion with human workers. Such an example of the collaborative robot is the YuMi robot, dual 7-DOF robot arms designed for precise manipulation of small parts better known in computer vision as rigid body. For further acceptance of such robots in the industry, some methods and sensors systems have to be developed to allow them to pick parts without the position of the part being known in advance, just as humans do. This thesis is focused on the implementation of an algorithm for determing the positition of the known parts. We first deal with a robot-camera calibration, then we propose a method to obtain the ground truth position of known parts. As step in between a 3D model of the known part needs to be created.

Keywords: manual, degree project, LaTeX

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

Abstrakt

Nová generace takzvaných spolupracujících robotů umožňuje použití malých robotických zbraní bez toho, aby byli izolováni od lidských pracovníků. Takovým příkladem spolupracujícího robota je robot YuMi, dvojitý 7-osý robot robotů určený pro přesnou manipulaci s malými částmi a dostupný v laboratoři Inteligentní a mobilní robotika CIIRC. Pro další přijetí takových robotů v průmyslu je třeba vyvinout některé metody a systémy snímačů, které by jim umožnily vybírat části bez předchozího znát umístění části, stejně jako lidé. Práce je zaměřena na implementaci algoritmu pro lokalizaci známých částí. Vedle lokalizace se část práce skládá z kalibrace kamery relativeley k robotovým a devolopingovým metodám pro získání pozemské pravdivé pozici dílů. . . .

Klíčová slova: manuál, závěrečnná práce, LaTeX

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 follow.

1.1 Motivation

For years. The industrial robot has undergoes through enormous development. Robot nowadays not only receives command from the computer. But also has the ability to make decision itself. Such abilities are well known in the world of the computer vision as recognizing and determining 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 method that goes from state-of-the-art to deep learning means where the object is usually represented with a CAD model or object's 3D reconstruction and typical task is detection of this particular object in the scene captured with RBGD or depth camera. Detection consider determining the location of the object in the input image. This is typical in robotics and machine vision applications where the robot usually does task like pick and place objects. However, localization and pose estimation is 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 requirement of real-time performance which make the the whole task even much more harder.

1.2 Goal

We attempt to provide an algorithm for determining the pose of a known parts similar to following pipeline "6D object pose estimation using RGBD data" [?]. In addition, a robot-camera calibration needs to be done, and a main requirement a 3D object model needed.

1.3 Thesis structure

The thesis consists of 5 chapters, **??** and **??**. The current chapter briefly describes the motivation and the goal for the part localization which we refer from here on through the whole thesis as 6D pose estimation of a rigid body in order to fit to the nomenclature giving in the perception field. Chapter 2 gives a background to camera calibration, openCV, open3D, ROS, prepossessing algorithm for segmenting the 3D image and related work about 6D pose estimation of the rigid body on which this work is building on. Chapter 3 describes the algorithms and the implementation for creating and collective ground data. Chapter 4 metric pair with the ground truth data. Chapter 5 concludes the thesis and showcase possible future works.

Chapter 2

Background

This chapter presents a briefly theoretical background and overview of the several API's and tools used in this thesis. The reader can skip this chapter, but it is good to know in order to better follow the thesis. To dive deeply in any topic described ahead, a reference is given.

2.1 RGB-D sensors

Nowadays novel camera systems like the Astra Orbbec and RealSense which provide both color and depth images became readily available. There are great expectations that such systems 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 camera are used.

Figure 2.1: RGB-D sensor(from Astra Orbbec documentation))

2.2 Camera Pinhole Model

There are many lens models but Pinhole camera is used in this thesis. Pinhole camera is the simplest device 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.

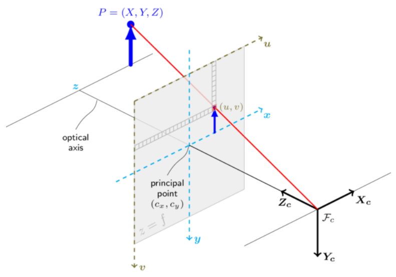

Figure 2.2: View of a Pinhole camera geometry (from Camera Calibration and 3D Reconstruction, openCV)

2.2.1 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 the Equation 3.1.

$$z_c * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K * \begin{bmatrix} \mathbf{R} \ \mathbf{t} \end{bmatrix} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.1}$$

2.2.2 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 3.3.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

2.2.3 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:

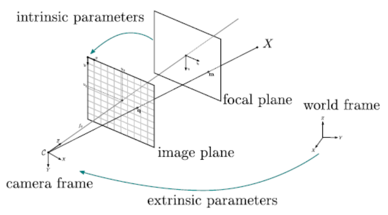

Figure 2.3: Overview of the transformation between the focal plane and the image plane

$$det(\mathbf{R}) = 1 \tag{2.3}$$

$$\mathbf{R}^T\mathbf{R} = I \tag{2.4}$$

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

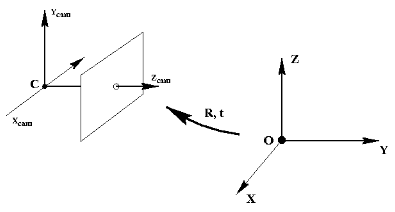

Figure 2.4: Overview of a world coordinate system and camera coordinate system

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.

$$P_c = \mathbf{R}.P_w + \mathbf{t} \tag{2.5}$$

The Equation 3.5 can also be written as:

$$P_c = [\mathbf{R}\ \mathbf{t}] \left[\frac{P_w}{1} \right] \tag{2.6}$$

2.3 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.

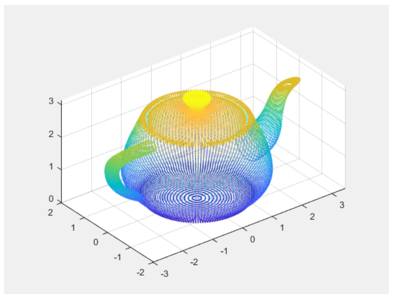

Figure 2.5: Overview of a point cloud (from MathWorks documentation)

2.4 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. However, topics are asynchronous, synchronous communication 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 [3].

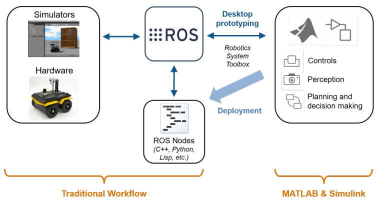

Figure 2.6: A ROS Overview

2.5 PCL

The PCL[4] 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

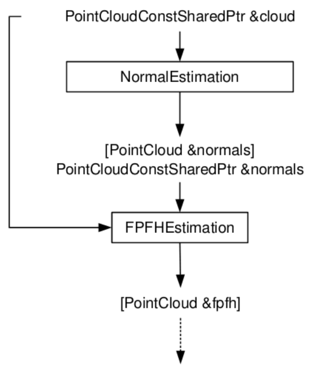

Figure 2.7: An example of the PCL implementation pipeline for Fast Point Feature Histogram (FPFH) [8] estimation.

2.6 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 [6], and multiway registration using pose graph optimization.

2.7 Tools for 3D data

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.

2.7.1 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.)[9],.

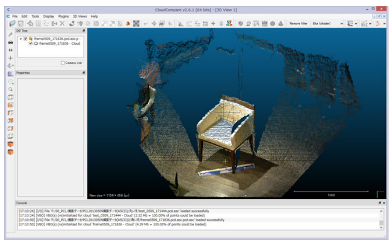

Figure 2.8: CloudCompare (view, edit and process).

2.7.2 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 [10].

2.7.3 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 model 3D objects with precision and control over modeling history [11].

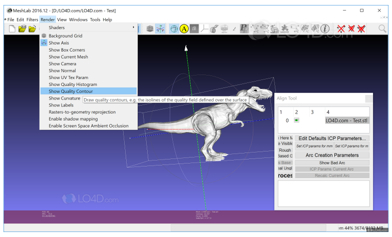

Figure 2.9: MeshLab (view, edit and process).

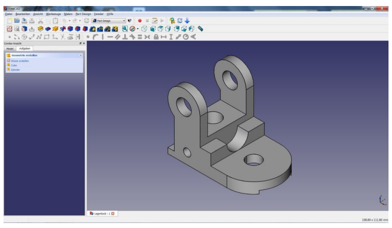

Figure 2.10: A view of the FreeCAD interface.

Chapter 3
Theory

This chapter presents a briefly theoretical background and overview of the several API's and tools used in this thesis. The reader can skip this chapter, but it is good to know in order to better follow the thesis. To dive deeply in any topic described ahead, a reference is given.

3.1 RGB-D sensors

Nowadays novel camera systems like the Astra Orbbec and RealSense which provide both color and depth images became readily available. There are great expectations that such systems 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 camera are used.

Figure 3.1: RGB-D sensor(from Astra Orbbec documentation))

3.2 Camera Pinhole Model

There are many lens models but Pinhole camera is used in this thesis. Pinhole camera is the simplest device 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.

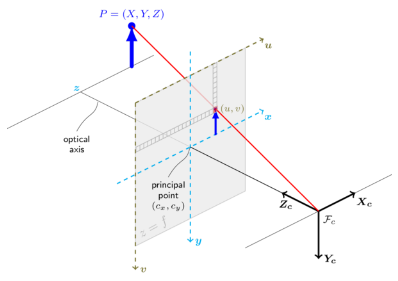

Figure 3.2: View of a Pinhole camera geometry (from Camera Calibration and 3D Reconstruction, openCV)

3.2.1 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 the Equation 3.1.

$$z_c * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K * \begin{bmatrix} \mathbf{R} \; \mathbf{t} \end{bmatrix} * \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.1}$$

3.2.2 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 3.3.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

3.2.3 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:

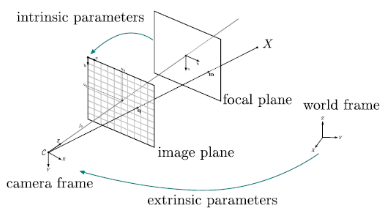

Figure 3.3: Overview of the transformation between the focal plane and the image plane

$$det(\mathbf{R}) = 1 \tag{3.3}$$

$$\mathbf{R}^T\mathbf{R} = I \tag{3.4}$$

Where I is the identity matrix. The matrix **R** and the vector **t** altogether are called camera's extrinsic parameters, see Figure.

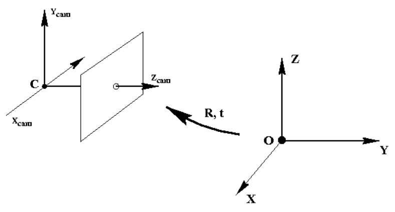

Figure 3.4: Overview of a world coordinate system and camera coordinate system

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.

$$P_c = \mathbf{R}.P_w + \mathbf{t} \tag{3.5}$$

The Equation 3.5 can also be written as:

$$P_c = [\mathbf{R}\ \mathbf{t}] \left[\frac{P_w}{1}\right] \tag{3.6}$$

3.3 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.

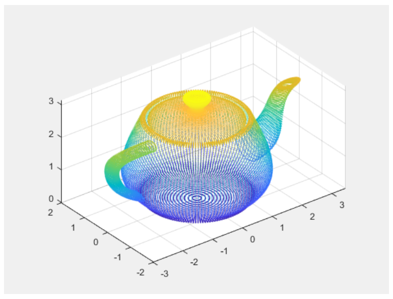

Figure 3.5: Overview of a point cloud (from MathWorks documentation)

3.4 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. However, topics are asynchronous, synchronous communication 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 [3].

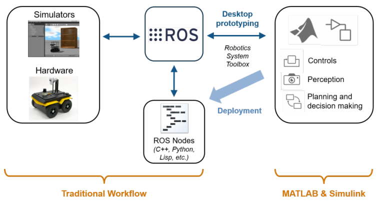

Figure 3.6: A ROS Overview

3.5 PCL

The PCL[4] 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

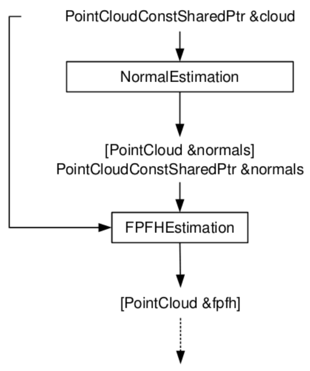

Figure 3.7: An example of the PCL implementation pipeline for Fast Point Feature Histogram (FPFH) [8] estimation.

3.6 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 [6], and multiway registration using pose graph optimization.

3.7 Tools for 3D data

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.

3.7.1 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.)[9],.

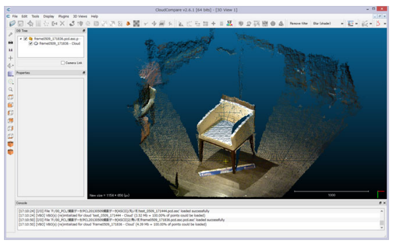

Figure 3.8: CloudCompare (view, edit and process).

3.7.2 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 [10].

3.7.3 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 model 3D objects with precision and control over modeling history [11].

Lorep ipsum [2]

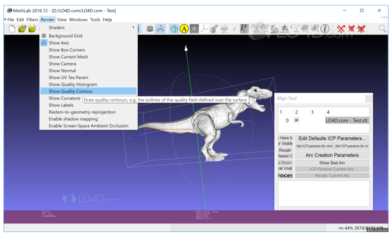

Figure 3.9: MeshLab (view, edit and process).

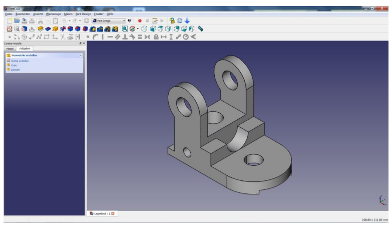

Figure 3.10: A view of the FreeCAD interface.

Bibliography

[1] J. Doe. *Book on foobar.* Publisher X, 2300.

[2] J. Doe. *Book on foobar.* Publisher X, 2300.

[3] 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`

[4] 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`

[5] 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/`

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

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

[8] 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.

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

[10] 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/`.

[11] FreeCAD, `https://www.freecadweb.org/`