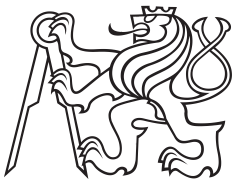
Master Thesis



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 posititon 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.

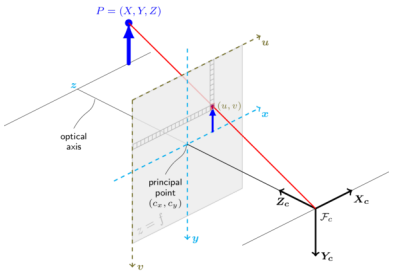


Figure 2.2: View of a Pinhole camera geometry (from Camera Calibration and 3D Reconstruction, openCV)

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 2.1.

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

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 2.3.

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

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:

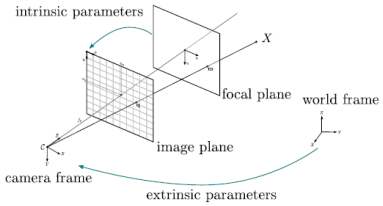


Figure 2.3: Overview of the transformation between the focal plane and the image plane

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

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

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

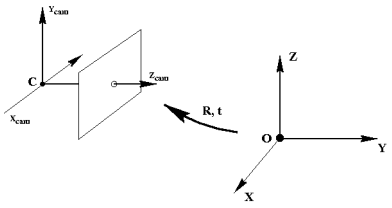


Figure 2.4: Overview of a world coordinate system and camera coordinate system

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 2.5 can also be written as:

$$P_c = [\mathbf{R}\ \mathbf{t}] \begin{bmatrix} P_w \\ \hline 1 \end{bmatrix} \tag{2.6}$$

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.

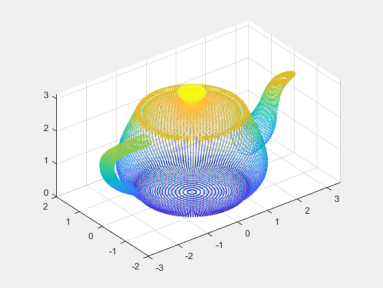


Figure 2.5: Overview of a point cloud (from MathWorks documentation)

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 [6].

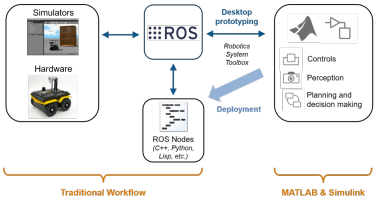


Figure 2.6: A ROS Overview

2.5 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

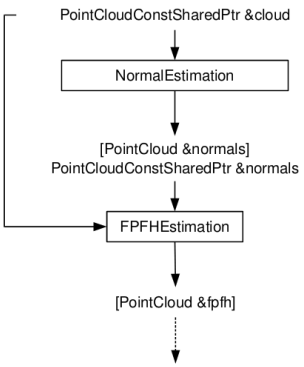


Figure 2.7: An example of the PCL implementation pipeline for Fast Point Feature Histogram (FPFH) [11] estimation.

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 [9], and multiway registration using pose graph optimization.

2.7 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.

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.)[12],.

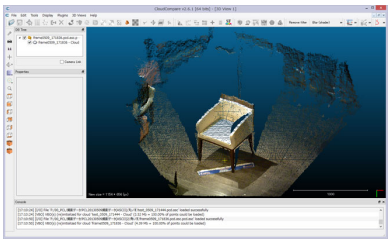


Figure 2.8: CloudCompare (view, edit and process).

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 [13].

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 [14].

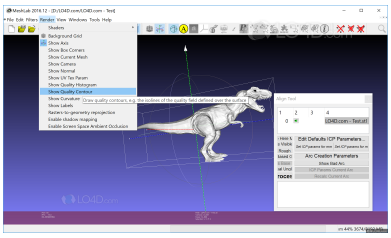


Figure 2.9: MeshLab (view, edit and process).

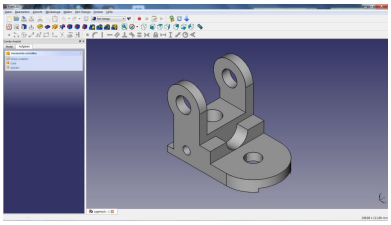


Figure 2.10: A view of the FreeCAD interface.

Chapter 3

Theory

This section presents the theory as well as the whole individual steps in details of the implemented system. The implemented system or pose estimation pipeline as we refer 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 as the offline stage where the CAD model is preprocessed, and the second one an online stage where the point cloud taken from the scence 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 stragy is applied where it outputs a 6-DOF pose estimation(ground truth) of the object.

3.1 Pose estimation pipeline

Using a local feature base method, the pose estimation pipeline is seen in Figure 3.1. The pipeline used in this thesis is inspired in [1], [2] and [3], with two major modifications, the first one is that the pipeline used in this thesis has the filtering part included in the preprocessing stage in order to better isolate the 3D object and the second modification is related to the matching strategy[15] where in most of the literatures, the preferred one is a hash table-based voting scheme, in contrast to the one used in this thesis, the hypothesize-and-test paradigm[15], e.g. RANSAC scheme, which is suitable in 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 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 feeding 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.

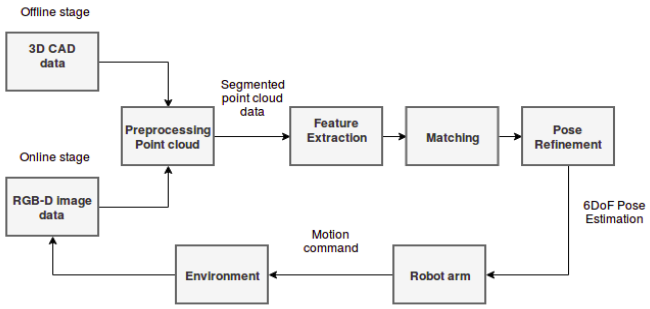


Figure 3.1: General architecture of proposed pose estimation pipeline

3.1.1 Preprocessing stage

After the filtering step, which is only applied to the point cloud represented the scence. The two clouds are downsampled with technique already implemented in the Point Cloud Library, such as Voxelgrid (VG) filter or Approximate Voxelgrid filtering. This step is required for speed 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 dominat plane in the image.

3.1.2 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 [16].

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 3.2. For more details and working examples the reader you should refer to [7]

Figure 3.2: 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. A further filtering is needed, such filtering is know as plane segmentation. And it is achievable with the RAMSAC based plane fitting method. RAMSAC method finds the largest set of points that fit to a plane. The plane equation in three dimensional point cloud can be defined as:

$$ax + by + cz + d = 0 \tag{3.1}$$

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 dataset and computes the parameters of the corresponding plane, after that it tries to enlarge the plane according to a given threshold,[18]. The step of segmenting the plane in order to remove it, is required condition for the subsequent use. Where a global registration is applied. The method is seen in Algorithm 1 and the flow chart

is seen in Figure 3.6

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 1: RANSAC for plane segmentation [3]

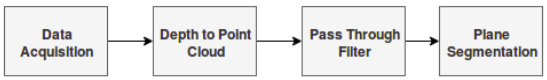


Figure 3.3: Flow Chart of Point Cloud Processing For Plane Segmentation

3.1.3 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, keypoint are found by sampling the point cloud or downsampling the cloud with VoxelGrid filter.

Voxel Grid filtering method [16] 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 3.4 to see the result of applying voxel grid. For more information, the reader should see [17]

Local surface feature description

Vertex normal estimation

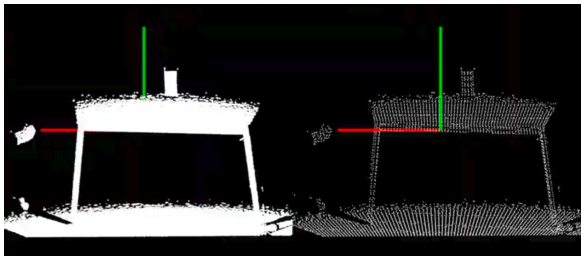


Figure 3.4: Left to Right: Input Image, Output after voxelGrid

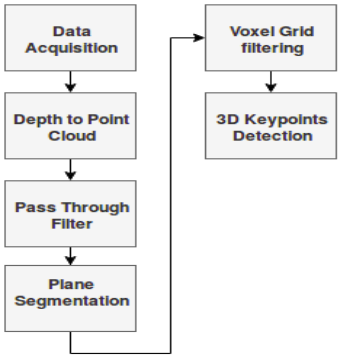


Figure 3.5: Flow Chart of Point Cloud Processing For Keypoints detection

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.

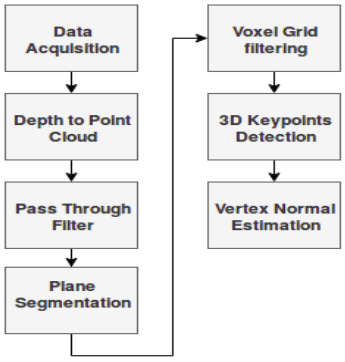


Figure 3.6: Flow Chart of Point Cloud Processing For Plane Segmentation

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 [19], classify 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 s be generating histograms according to the geometric attributes(e.g.,normals) of the local surface.

Figure 3.7: Flow Chart of Point Cloud Processing For Feature Detection

Fast Point Feature Histogram
Fast Point Feature Histogram (FPFH) in [20], is a developed version of the Point Feature Histogram (PFH) [20] with a reduced computational complexity and 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:

$$\begin{aligned} \alpha &= v \cdot n_j \\ \phi &= \frac{u \cdot (p_j - p_i)}{|p_j - p_i|} \\ \theta &= \arctan(w \cdot n_j, u \cdot n_i) \end{aligned} \tag{3.2}$$

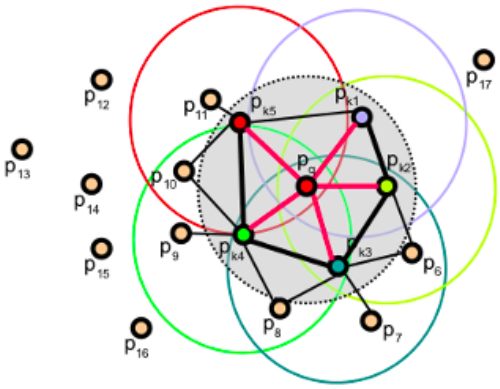
The computation above is called a Simplified Point Feature Histogram (SPFH) which is binned by three angular variations (α, ϕ, θ), then in the second step, the FPFH of each point is computed using both the SPFH of itself and the weighted ones of its neighbours as follow:

$$FPFH(p) = SPFH(p) + \frac{1}{k} \sum_{n=1}^{k} \frac{1}{\omega_k} SPFH(p_k) \tag{3.3}$$

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

3.1.4 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 3.8: The influence region diagram for a Fast Point Feature Histogram. [20]

Therefore, a searching strategy is needed in order to find the proper point correspondences 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. In [15] 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)[15] 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 3.9: Flow Chart of Point Cloud Processing For Finding Correspondence

3.1.5 Local refinement

The last step of the pipeline is the so-called refinement of the alignment achieved by a coarse matching generated in 3.1.4. 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, find 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 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 2. For more details about the ICP and its variants, the reader should refer [15] and [21]

Standard ICP is seen in Algorithm kkkk.

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 2: Standar ICP

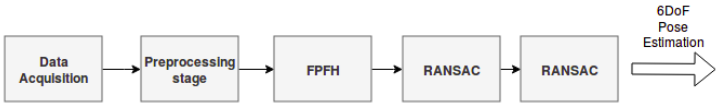


Figure 3.10: Flow Chart of The Pose Estimation Pipeline

3.1.6 Local refinement

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

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

[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] 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.

[16] 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/.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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