

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Recognition and 6 DoF Pose Estimation of 3D Models in Depth Sensor Data

Inês de Sousa Caldas



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Armando Jorge Miranda de Sousa (PhD)

Co-Supervisor: Carlos Miguel Correia da Costa (MSc)

October 11, 2018



# **Recognition and 6 DoF Pose Estimation of 3D Models in Depth Sensor Data**

**Inês de Sousa Caldas**

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Nuno Cruz (PhD)

External Examiner: Gil Lopes (PhD)

Supervisor: Armando J. Sousa (PhD)

October 11, 2018



# Abstract

Automation of repetitive tasks using robots has revolutionized industrial manufacturing for a very long time, allowing mass customization of a wide range of products at reduced cost, improved quality and flexibility. However, with manufacturing lines still under-optimized, the European project Scalable4.0 aims to develop and demonstrate an Open Scalable Production System Framework (OSPS) that enables optimization and maintenance of production lines ‘on the fly’, through visualization and virtualization of the line itself. Within the scope of the Scalable4.0 project, the research in this thesis is aimed at facilitating the integration of a global features pipeline for 3D object recognition and pose estimation in a manufacturing environment. With the increasing accuracy and sampling rate of the new depth sensors, they’re specially suitable to be used in manufacturing lines and comply with the accuracy constraints.

The goal of this thesis is to develop and study a pipeline that uses global features to recognize rigid objects from a single viewpoint and estimate their position and orientation in the real world. The objects used to train the system are represented as 3 dimensional meshes, and the real objects are sensed using a depth sensor.

Because descriptors are the most important element for robust object recognition system as they assign a unique identification to each object that withstands pose and illumination variations, the system implements various global descriptors available in Point Cloud Library (PCL) and are configurable within the Robot Operating System (ROS) package. The descriptors from the training set are matched to the descriptors of a scene depth image to find the 3 dimensional pose of the model in the scene. The pose estimation is then refined iteratively using a registration method.

The pipeline was tested using a dataset with 5 objects provided by the industrial partners of the Scalable4.0 project, PSA and Simoldes. To capture the environment of the test scene we use the industrial structured light sensor Photoneo PhoXi® 3D Scanner S.

Our experiments have proven the system is capable of segmenting the scene in clusters, each one representing an appropriate candidate to recognize. The evaluation shows also the importance of choosing the proper descriptor for the dataset in question. In our results the Viewpoint Feature Histogram (VFH) and Ensemble of Shape Functions (ESF) descriptors proved to be able to recognize the objects, whereas the Clustered Viewpoint Feature Histogram (CVFH) descriptor didn’t. The iterative closest point registration method estimated the poses with an error no greater than  $2 * 10^{-5}m$ .



# Resumo

A automação de tarefas repetitivas usando robots tem vindo a revolucionar o sector industrial há bastante tempo, permitindo a customização em massa de uma ampla gama de produtos a um custo reduzido, de grande qualidade e flexibilidade. Contudo, com muitas das linhas industriais ainda sub-otimizadas, surge o projeto europeu Scalable4.0 cujo objetivo é desenvolver uma *framework*, a *Open Scalable Production System Framework* (OSPS), que permita a otimização e manutenção das linhas de produção no momento através da visualização e virtualização das linhas de produção. No contexto deste projeto, esta dissertação pretende facilitar a introdução de uma *pipeline* que usa características globais para o reconhecimento de objetos 3D e a estimação da sua pose, num ambiente industrial.

O nosso objetivo consistiu em desenvolver um *Robot Operating System* (ROS) *package* customizável com um sistema de reconhecimento que usa descritores globais capaz de reconhecer objetos rígidos a partir de um ponto de vista e estimar a sua posição e orientação no mundo real. Os objetos usados para treinar o sistema provém de modelos 3D codificados usando o software de Desenho Assistido por Computador ou CAD (do inglês *Computer Aided Design*), enquanto que para testar o sistema, usou-se nuvens de pontos capturadas usando sensores de profundidade.

Como os descritores são o elemento mais importante para garantir a robustez do reconhecimento de objetos, dado que identificam de forma única cada objeto, o sistema de reconhecimento permite o uso vários descritores globais disponível na biblioteca *Point Cloud Library* (PCL). Para o reconhecimento, é feita a correspondência dos descritores do conjunto de treino com os descritores de uma imagem com informação de profundidade para permitir calcular a pose 3D do modelo na cena. Encontrada a pose esta é refinada iterativamente usando um método de registro.

A pipeline foi testada com um conjunto de 5 objetos fornecidos pelos parceiros industriais do projeto Scalable4.0, a PSA e a Simoldes. As cenas de teste foram capturadas usando um sensor de luz estruturada, o Photoneo PhoXi® 3D Scanner S.

As nossas experiências revelaram que o sistema é capaz de segmentar apropriadamente a cena em grupo de pontos cada um representando um objeto candidato a ser reconhecido. Os nossos resultados provaram também a importância de escolher um descritor apropriado às características dos objetos a analisar. Para os objetos usados nesta dissertação, os descritores Viewpoint Feature Histogram (VFH) e *Ensemble of Shape Functions* (ESF) mostraram ser capazes de classificar os objetos, enquanto o descritor *Clustered Viewpoint Feature Histogram* (CVFH) não. A estimação da pose usando o descritor *Camera Roll Histogram* (CRH) e o algoritmo *Iterative Closest Point* (ICP) estimaram a pose dos objetos com um erro não superior a  $2 * 10^{-5}m$ .



# Acknowledgements

To my supervisors for guiding me when help was necessary.  
To my family for always loving and supporting me.

Inês de Sousa Caldas



*“Tudo é ousado para quem nada se atreve”*

Fernando Pessoa



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	2
1.2	Goals . . . . .	2
1.3	Dissertation Outline . . . . .	2
<b>2</b>	<b>Related Work</b>	<b>5</b>
2.1	Object Recognition . . . . .	5
2.1.1	Segmentation and Clustering . . . . .	5
2.1.2	Detection Methods . . . . .	6
2.1.3	Keypoint selection . . . . .	7
2.1.4	Recognition Pipelines . . . . .	11
2.2	Registration . . . . .	11
<b>3</b>	<b>Relevant software / hardware technologies</b>	<b>13</b>
3.1	Software . . . . .	13
3.1.1	OpenCV . . . . .	13
3.1.2	PCL . . . . .	13
3.1.3	ROS . . . . .	14
3.2	Camera Systems . . . . .	14
3.2.1	Time of Flight System . . . . .	14
3.2.2	Structured Light System . . . . .	15
<b>4</b>	<b>Recognition and 6 DoF Pose Estimation System</b>	<b>17</b>
4.1	System Overview . . . . .	17
4.2	Data Structures for Efficient Searches in Point Clouds . . . . .	17
4.2.1	Octrees . . . . .	17
4.2.2	k-d trees . . . . .	18
4.3	Normal Estimation . . . . .	19
4.4	Downsampling and Filtering . . . . .	19
4.4.1	Voxel Grid Sampling . . . . .	19
4.4.2	Passthrough Filter . . . . .	20
4.4.3	Radius Outlier Removal Filter . . . . .	20
4.4.4	Statistical Outlier Removal Filter . . . . .	20
4.5	Training Phase . . . . .	21
4.6	Testing Phase . . . . .	21
4.6.1	Segmentation . . . . .	21
4.6.2	Features Description and Matching . . . . .	23
4.6.3	Camera Roll and 6-DoF Pose . . . . .	24

## CONTENTS

4.6.4	Post Processing . . . . .	24
<b>5</b>	<b>Experiments and Results</b>	<b>26</b>
5.1	The Dataset . . . . .	26
5.2	Experiments . . . . .	27
<b>6</b>	<b>Conclusions and Future Work</b>	<b>34</b>
	<b>References</b>	<b>36</b>
<b>A</b>	<b>The Dataset</b>	<b>38</b>
<b>B</b>	<b>Photoneo PhoXi® 3D Scanner S</b>	<b>41</b>

# List of Figures

2.1	Typical result of detecting obstacles and objects in a table top setting (a). Even obstacles (b, red) that, in 3D, do not stick out of the supporting surface (b, green) like the red lighter are perceived. Detected objects (c) are randomly coloured. For being able to grasp an object, the respective cluster is not considered as an obstacle (in this example the Pringles box). [HHRB12]	6
2.2	Darboux frame between a point pair [Rus09]	8
2.3	Object recognition pipelines suggested by [AMT <sup>+</sup> 12].	11
3.1	Point cloud library	14
3.2	The principle of ToF depth camera: The phase delay between emitted and reflected IR signals are measured to calculate the distance from each sensor pixel to target objects [HLCH12]	15
3.3	PhoXi 3D Scanner S	15
3.4	Structured Light Principle	16
4.1	Octree of a bunny mesh model.	18
4.2	The four steps to create a 2D k-d tree.	18
4.3	Point cloud of a table before (left) and after (right) voxel grid downsampling.	20
4.4	Point cloud of a table before (left) and after (right) statistical outlier removal.	21
4.5	The Stanford bunny mesh model and two generated views from the same view-point but different resolutions.	22
4.6	Simplified 3D object recognition pipeline	23
4.7	A segmentation of a table scene and the object clusters found to lie on it from an unorganized point cloud dataset. [RBMB09]	24
5.1	Two views from the full generated <i>tray</i> pointcloud.	26
5.2	Two views from the full generated <i>filter</i> pointcloud.	27
5.3	Two views from the full generated <i>cap</i> pointcloud.	27
5.4	Two views from the full generated <i>8080</i> pointcloud.	27
5.5	Two views from the full generated <i>8537</i> pointcloud.	27
5.6	Picture of the test scene used in the experiments.	28
5.7	Point cloud of the test scene after the preprocessing step (left) and after the extraction of the dominant plane (right).	28
5.8	The computed clusters by the euclidean clustering algorithm.	29
5.9	Pose estimations using ESF descriptors.	33
A.1	Two views from the <i>tray</i> .	38
A.2	Two views from the <i>filter</i> object.	39
A.3	Two views from the <i>cap</i> object.	39

## LIST OF FIGURES

A.4	Two views from the <i>8080</i> object.	40
A.5	Two views from the <i>8537</i> object.	40

# List of Tables

5.1	The confusion matrix for the clusters classification using VFH descriptors . . . .	29
5.2	The confusion matrix for the clusters classification using ESF descriptors . . . .	30
5.3	The confusion matrix for the clusters classification using CVFH descriptors . . . .	30
5.4	Number of points of the dataset models views and the clusters for determining the rigid transformation between the using the ICP algorithm. . . . .	30
5.5	MSE for the <i>tray</i> pose estimations. . . . .	31
5.6	MSE for the <i>filter</i> pose estimations. . . . .	31
5.7	MSE for the <i>cap</i> pose estimations. . . . .	31
5.8	MSE for the <i>8080</i> pose estimations. . . . .	31
5.9	MSE for the <i>8537</i> pose estimations. . . . .	32

## LIST OF TABLES



# Abbreviations

3D	3 Dimensional
3DSC	3D Shape Contexts
CAD	Computer Aided Design
CVFH	Clustered Viewpoint Feature Histogram
DoF	Degrees of Freedom
ECT	Ensemble Collaborative Trackers
ESF	Ensemble of Shape Functions
FLANN	Fast Library for Approximate Nearest Neighbors
FPFH	Fast Point Feature Histogram
FPS	Frames Per Second
HCI	Human Computer Interaction
Hz	Hertz
ICP	Iterative Closest Point
IK	Inverse Kinematics
IR	Infrared
ISS	Intrinsic Shape Signatures
MSE	Mean Square Error
NN	Nearest Neighbours
PCA	Principal Component Analysis
PCL	Point Cloud Library
PFH	Point Feature Histograms
PSO	Particle Swarm Optimization
RANSAC	Random Sample Consensus
RGB-D	Red, Green, Blue - Depth
ROI	Region of Interest
ROS	Robot Operating System
SDC	Shape Distribution Component
SHOT	Signature of Histograms of Orientations
SIFT	Scale-Invariant Feature Transform
SUSAN	Smallest Univalued Segment Assimilating Nucleus
ToF	Time of Flight
VFH	View point Feature Histogram

# Chapter 1

## Introduction

The understanding of the observed environment based on computer registered images is one of the most profound problems and goals faced by the vision community and other research areas dependent on it, such as robotics. In particular, finding the number, the type, the properties and finally the pose of objects is a very complex problem. The analysis and interpretation of images and extraction of key information is intuitive, effortless and instantaneous for humans, however it is one of the crucial capabilities that computer systems still lack today. Despite the various attempts to mimic human vision capabilities, computer algorithms in this field still face many challenges due to the complexity of the methods and superficial knowledge of its progress in the human brain. One of the key problems associated with the manipulation of objects is their detection, recognition and localization in the visual scene. The latter task seems to be particularly difficult, however, it became solvable in nearly real time when sensors like Microsoft Kinect appeared in the market. The generated images by these sensors does not only contain the usual three color components of the observed scene for each pixel, but it also holds the distances of the observed points from the sensor. This opens a whole new range of possibilities for analysis and processing of information, but at the same time, it creates new challenges that require new solutions to deal with more complex data.

Whereas traditional methods work with 2D images and typically describe pose of the detected object by a bounding box, which encodes 2D translation and scale usually cannot provide information about the object orientation and a precise motion of the object distance, when the surface geometry of the object is complex, algorithms based on depth data facilitate the computation of the pose of a rigid object. A 6DoF pose encodes 3 degrees in translation and 3 in rotation, and its full knowledge is required in many robotic applications, for example, 6D object pose facilitates spatial reasoning and allows an end-effector to act upon an object.

## 1.1 Context

Automation of repetitive tasks using robots has revolutionized industrial manufacturing for a very long time, allowing mass production of a wide range of products at reduced cost and improved quality. Some of these tasks required accurate and in real-time 3D object recognition along with a 6DoF pose estimation. Since the launch of the Microsoft Kinect, a variety of new sensors have appeared, specially targeted for robotic systems. With the increasing accuracy and sampling rate of the new sensors, they're specially suitable to be used in manufacturing lines and comply with the accuracy constraints.

With manufacturing lines still under-optimized, due to decisions being made without taking into consideration the overall picture of the production floor, usually this leads to sub-optimal decisions. Scalable4.0<sup>1</sup> is an European research projects that aims to develop and demonstrate an open scalable production system framework (OSPS) that enables optimization and maintenance of production lines 'on the fly', through visualization and virtualization of the line itself. The outcome of the project focus on line monitorization, control and construction in real time. This project has two industrial partners (PSA and Simoldes), with demanding use cases related with manufacturing and assembly of products. Its main goal is the development of a multi-product production line in the group, capable of efficiently handling demand variations using adjustable and flexible workstations with intuitive reprogramming interfaces.

## 1.2 Goals

The research in this thesis aims to facilitate the integration of a global features pipeline for 3D object recognition in a manufacturing environment. The manufacturing pipelines are often controlled environments which can facilitate the object's recognition on top of the conveyor. The system must be able to accurately determine the identity, position, and orientation of objects in the workspace. This overall criterion includes minimum levels of accuracy, precision, and robustness to different environments and tasks, as well as the requirement that the automation system is efficient enough to complete tasks quickly enough to be useful in complex workstations. The system was tested with a DataSet from real parts provided by the industrial partners of the Scalable4.0 project and a structured light sensor.

## 1.3 Dissertation Outline

The remaining of this document is split over 5 chapters. This document is structured as following:

- In chapter 2 a revision and summary of state-of-the-art methodologies for 3D object recognition is done.

---

<sup>1</sup><https://www.scalable40.eu/>

## Introduction

- In chapter 3 we summarize the existing and most relevant technologies for work done in this thesis.
- In chapter 4 the architecture of the developed system is described;
- In chapter 5 the experiments to evaluate the system and the obtained results are exposed and discussed;
- In chapter 6 some final considerations about the work done are made and future research topics are presented.

## Introduction

## Chapter 2

# Related Work

### 2.1 Object Recognition

#### 2.1.1 Segmentation and Clustering

Segmentation methods aim to divide large amounts of data into smaller subsets that can be processed faster and regrouping elements that share the same properties together. These two aspects are of great importance for processing 3D data, where is necessary to tackle large point cloud datasets which can be computationally demanding. Moreover, data segmentation can also be applied to seek for certain patterns or structures in a point cloud dataset, as pointed out by Rusu [Rus13]. For example, a set of points can be replaced by any geometric primitives detected (planes, spheres, ...) and thus regrouping the points into clusters in order to simplify the dataset's treatment.

The simpler methods for dividing point cloud datasets into clusters are based on spatial decomposition techniques. These methods group data together according to their proximity, namely by the Manhattan or Euclidean distance between points. Due to noise or errors in the dataset, the true nearest neighbour of a point could be different than the one with the nearest distance, for that reason sometimes a Kd-structure for finding nearest neighbours is applied prior to the clustering. The euclidean clustering algorithm can be extended to include additional information of neighbouring points, such as color and information about its surrounding geometry. For example, Holz *et al.* [HHRB12] proposed a system that allows a robot to detect obstacles and grasp objects, based on segmentation and classification of planes using a RGB-D camera. The first step, consists in compute surface normals of points taken from integral images. Next, points with similar local surface normals orientations are clustered together as potential set of planes. Assuming that all points in a cluster lay in the same surface, the averaged and normalized surface normal is computed and used to estimate the distance from the origin to the plane. With frame rates up to 30Hz, the authors are able to detect obstacles or objects for manipulating tasks. An example of the results obtained by this approach is showed on figure 2.1. In the case of point cloud datasets obtained from multiple

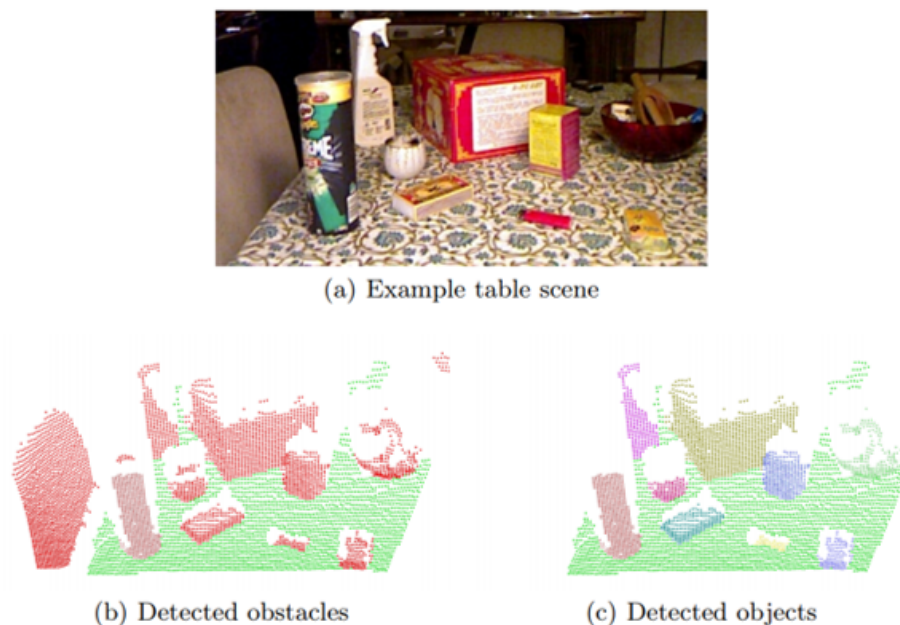


Figure 2.1: Typical result of detecting obstacles and objects in a table top setting (a). Even obstacles (b, red) that, in 3D, do not stick out of the supporting surface (b, green) like the red lighter are perceived. Detected objects (c) are randomly coloured. For being able to grasp an object, the respective cluster is not considered as an obstacle (in this example the Pringles box). [HHRB12]

RGB-D sensors, Susanto *et al.* [SRS12] perform the segmentation process on each camera independently, therefore obtaining a segmented view for each sensor. The registration step is applied next. This approach allows to minimize camera calibration inaccuracy effects or any problem resulting from the different field-of-views of each sensor. As a result, the clustering algorithm has to be executed for each camera hence consuming more computational resources.

### 2.1.2 Detection Methods

Object detection implies the comparison of different set of points or templates in order to find correspondences between a model and a query image. The question is which parameters give a unique representation of an object and should be taken into account? The cartesian coordinates and Euclidean metrics do not provide with sufficient information for point to point correspondence, additional information has to be considered. The structure of information that uniquely describes a point or template is known as *descriptor* or *point feature representation*. Rusu [Rus13] claims that a good point descriptor should be invariant to rigid transformations, varying sampling density and noise. Descriptors can be divided into two distinct categories. In one category are the so called global descriptors which are computed on large selection of points. Generally they are not centred around a specific point, but instead describe a entire object. On the other end, local descriptors are centred on a specific point and they contain information from a neighbourhood which is usually determined by a selection of point within a radius of the centre point. Usually, they also contain

background information that helps to identify points of interest. We will describe some available descriptors.

### 2.1.3 Keypoint selection

Keypoints are spatial locations, or points in a image that contain interesting geometry for computing a useful descriptor. The purpose of keypoint selection is to reduce the total number of points to only the ones that contain the most relevant information. The selection of good keypoints is critical to achieve well performing object detection and registration when working with point clouds. This is because most features calculated for the point cloud are based on keypoints, and not the full point cloud captured with a sensor. Features are descriptive properties that are used to identify a particular region or area of a point cloud.

The most commonly used keypoint detectors are:

- Harris3D ([[HS88](#)])
- SIFT3D ([[RC11](#)])
- SUSAN ([[SB97](#)])
- ISS3D ([[Zho09](#)])

Filipe and Alexandre [[FA14](#)] presented an explanation and evaluation of the current key point detectors available to be used in the PCL. They evaluated the invariance of the 3D key point detectors depending on the translation, scale, and rotations changes. They used the relative and absolute repeatability rate as criteria for performance evaluation. Their experiments showed that overall, SIFT3D and ISS3D achieved the best scores in terms of repeatability, and that the ISS3D proved to be the most efficient.

The Scale Invariant Feature Transform (SIFT) keypoint detector was proposed in [[Low04](#)]. The modified algorithm used on 3D data sets was presented by Rusu and Cousins in [[RC11](#)]. The most notable difference between the two algorithms are that SIFT3D uses a 3D version of the Hessian to select interest points, and that the intensity of a pixel is changed to the principal curvature of a given point.

#### 2.1.3.1 Local Methods

A local descriptor is computed for each detected keypoint and has region of influence defining a neighbourhood. If this neighbourhood is too small, the descriptor will only describe basic features like planes and corners and will lose its discriminability. On the other hand, if its too large, it will contain much information from the background and it will not match to anything. The robustness of local feature descriptors is dependent of the quality of the sensor data and the overall size of objects along with the complexity of their geometry.

### The Point Feature Histograms (PFH)

The PFH descriptor was developed in 2008 [RBMB08]. The descriptor's goal is to generalize both the surface normals and the curvature estimates. Apart from the usage for point matching, the PFH descriptor is used to classify points in a point cloud, such as points on an edge, corner, plane, or similar primitives. A Darboux frame (see Figure 2.2) is constructed between all point pairs within the local neighbourhood of a point  $p$ . The source point  $p_s$  of a particular Darboux frame is the point with the smaller angle between its normal and the connecting line between the point pair  $p_s$  and  $p_t$ . The Darboux frame  $u, v, w$  is defined as:

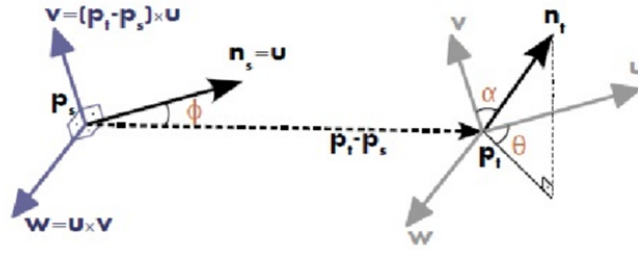


Figure 2.2: Darboux frame between a point pair [Rus09]

$$u = n_s \quad (2.1)$$

$$v = u \times \frac{p_t - p_s}{\|p_t - p_s\|} \quad (2.2)$$

$$w = u \times v \quad (2.3)$$

Three angular values  $\beta, \phi$  and  $\omega$  and one distance value  $d$  describe the relationship between the two points and the two normal vectors, and are computed as follows:

$$\beta = v \cdot n_t \quad (2.4)$$

$$\phi = u \cdot (p_t - p_s) / d \quad (2.5)$$

$$\theta = \arctan(w \cdot n_t, u \cdot n_t) \quad (2.6)$$

$$d = \|p_t - p_s\| \quad (2.7)$$

These four values are added to the histogram of the point  $p$ , which shows the percentage of point pairs in the neighbourhood of  $p$ , which have a similar relationship. The multi-dimensional his-

## Related Work

togram provides an informative signature for the feature representation, which is invariant to the 6D pose of the underlying surface and is able to handle different sampling densities or noise levels present in the neighbourhood.

### **Fast Point Feature Histogram (FPFH)**

The FPFH variant [RBB09] is used to reduce the computation time at the cost of accuracy. It discards the distance  $d$  and decorrelates the remaining histogram dimensions. Therefore, FPFH uses a histogram with only  $3b$  bins instead of  $b^4$ , where  $b$  is the number of bins per dimension. The time complexity is reduced by only considering the direct connections between the current keypoint and its neighbours, removing additional links between neighbours. This descriptor relies on the center point having a well-defined normal and, conceivably, a well-defined curvature. The latter, may not be defined in case we are dealing with shapes where the curvature is similar in all directions, like planes or bowls. The FPFH descriptor is invariant to rigid transformation and, even if the principle curvature direction is not well defined, the descriptor can still be computed and becomes invariant to rotation about the normal of the anchor point. So two of these degenerate features can still be matched and be aligned to each other up to an unknown rotation over their normals.

### **Spin Image**

A spin image descriptor [JH98] is computed in a cylindrical coordinate system, with the centre point and normal associated with that point used to orient the spin image. The descriptor defined by the reference point and its normal, is computed by calculating the histogram of radial and elevation distances of the point's neighbours. The resulting histogram is 2D and can be perceived as an image which spins around the normal of the reference point to accumulate points in its bins; hence the name spin image. Due to integration around the normal of the reference point, spin images are invariant to rigid transformations and can be matched by comparing corresponding bins. Despite the fact this descriptor has a limited discriminative ability, is computationally cheap and can be useful for certain applications.

### **Signature of Histograms of Orientations (SHOT)**

The SHOT descriptor [TSDS10] is an attempt of the SIFT descriptor but for the 3D cases. This descriptor encodes information about the topology within a spherical support structure. A local repeatable reference frame is produced using the support points. Next, the support points are placed into a set of subdivided spherical bins. The histogram orientations are taken for each bin. As the points move from bin to bin, they tend to decidedly change, there is an additional set of quadrilinear interpolation when placing the points into the bins. Finally, when all local histograms have been computed, they are stitched together into a final descriptor. The use of a local frame makes the SHOT descriptor invariant to rotation.

### 3D Shape Contexts (3DSC)

The 3D Shape Context descriptor [FHK<sup>+</sup>04] is like the SHOT descriptor illustrated above, however it ignores the normals within each bin and lacks the sophistication of the previous method for computing the local reference frame. Apart from also using a spherical grid on each of the keypoints, it simply uses the normal to align the north pole of the binning systems. The grid consists of bins along the radial, azimuth and elevation dimensions. The divisions along the radial dimension are logarithmically spaced and each bin makes a weighted count of the number of points that fall into it. The weights used are inversely proportional to the bin volume and the local point density.

### 2.1.3.2 Global Descriptors

Unlike local descriptors, global descriptors encode the object geometry. They are not computed for individual points, but for a whole cluster that represents an object. For that reason, a preprocessing step is necessary to retrieve possible object clusters. Because they have a support region defined as input, generally determined by a segmentation algorithm or by using an entire scan depending on the problem, these descriptors can be very useful if the segmentation is easy, however in most cases segmentation is not an easy problem. Regardless, given a cluster of points, the task of the descriptor is to compute a high dimensional representation of the set of points. Ideally, this representation is robust to occlusions (missing points) and changes in density from scanning patterns and point of view. Both [AVB<sup>+</sup>11] and [WV11] have compared global descriptors. In the existing research, less attention has been given to quantitatively comparing the pipeline based on global descriptors with the pipeline based on local descriptors.

### Viewpoint Feature Histogram (VFH)

The VFH adds viewpoint variance to the FPFH by using the viewpoint vector direction. The descriptor consists of two components: a viewpoint component and an extended FPFH component. Rusu *et al.* [RBTH10] estimate a centroid and average normal for the collection of points; compute the FPFH using the centroid and average normal as the anchor point; and finally add a viewpoint component. The viewpoint component is defined as the angle between the centroid of the object in the sensors coordinate frame and the normal of the support point. The histogram computed from the viewpoint component and the histogram computed from the spatial component are concatenated into a larger histogram which bins are then normalized. Although VFH shows promise in detecting objects along with pose, it has some limitations. For example, VFH is particularly sensitive to occlusions because missing points will change the number of bins in the histogram which will then be normalized without them. Also, VFH requires a lot training with examples from all poses of the object to be recognized implying a greater memory usage.

### The Clustered Viewpoint Feature Histogram (CVFH)

The Clustered Viewpoint Feature Histogram (CVFH) descriptor for a given point cloud dataset containing 3D data and normals, was proposed in [AVB<sup>+</sup>11] since the original VFH is neither robust to occlusion or other sensor artefacts, or to measurements errors. This descriptor expands

on the VFH with an additional segmentation step. Instead of computing a single VFH histogram, the object is firstly split into  $k$  stable locally smooth regions by first removing points with high curvature and then applying a smooth region-growing algorithm. This algorithm enforces several constraints on the points belonging to each region, such as restrictions on permitted distances and differences of normals. Then, a VFH is computed for every region. Additionally, a Shape Distribution Component (SDC) is also computed and included to each histogram. This additional step, encodes information about the distribution of the point's in the given region's centroid and allows to distinguish objects with similar traits.

### Ensemble of Shape Function (ESF)

The ensemble of shape function (ESF) descriptor introduced in [WV11] is an ensemble of ten 64-bin-sized histograms of shape functions describing the properties of the point cloud. The shape functions consist of angle, point distance, and area shape functions. A voxel grid serves as an approximation of the real surface and is used to separate the shape functions into more descriptive histograms representing point distances, angles, and areas, either on the surface, off the surface, or both. Contrary to the other global descriptors, the ESF does not require the use of surface normals to describe the cloud. The advantage of the ESF descriptor is that it can be efficiently calculated directly from the point cloud with no preprocessing necessary, as this descriptor is robust to sensor noise and incomplete surfaces.

#### 2.1.4 Recognition Pipelines

In [AMT<sup>+</sup>12] the authors present two recognition pipelines, one based on local descriptors and the other based on global descriptors. The different steps in each pipeline are depicted in figure 2.3.

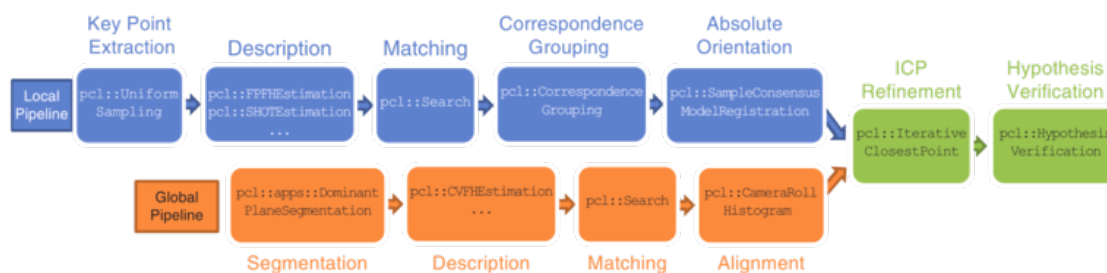


Figure 2.3: Object recognition pipelines suggested by [AMT<sup>+</sup>12].

## 2.2 Registration

Registration is the process of finding a spatial transformation that aligns two point clouds. The iterative closest point (ICP) algorithm, introduced in [BM92], is one of the most popular algorithms for registration of point clouds. This generic algorithm, in addition to point clouds, is able

## Related Work

to register the source cloud to some other representations of the target geometric data, such as line segment sets, triangle sets, curves or surfaces. The only condition is that it must be possible to find a target point that has the smallest Euclidean distance from the selected source point. It is possible to register both, 2D and 3D geometric data with the ICP. In addition to that, a convergence of this algorithm was proven in [BM92]. The main disadvantage of the ICP algorithm is a fact that it converges only to a local optima. It means that if the searched transformation between the target and source clouds is not small enough, the algorithm can compute a wrong transformation.

## Chapter 3

# Relevant software / hardware technologies

Object tracking in complex scenes, is a multidisciplinary problem that requires advanced computer software systems. In order to benefit the development and employment of the tracking system, several frameworks and libraries can be of great help. Amid the most relevant are Open Source Computer Vision Library (OpenCV) for computer vision applications, Point Cloud Library (PCL) for point cloud processing, Gazebo for simulation and testing and ROS for the system architecture.

### 3.1 Software

#### 3.1.1 OpenCV

OpenCV <sup>1</sup> is a computer vision library with state-of-the art algorithms that can be used to detect and identify objects, track camera movements, track moving objects, extract 3D models of objects and produce 3D point clouds from stereo cameras. It has C++, C, Python, Java and Matlab interfaces.

#### 3.1.2 PCL

The PCL <sup>2</sup> is a large scale, open source project for 2D/3D image and point cloud processing. The PCL framework contains a large number state-of-the art algorithms that can be used for filtering, feature estimation, surface reconstruction, point clouds registrations and also to perform object segmentation, recognition and tracking. Figure 3.1 gives an overview of the main modules available in PCL.

---

<sup>1</sup><https://opencv.org/>

<sup>2</sup><http://pointclouds.org/>

<sup>3</sup><http://pointclouds.org/about/>

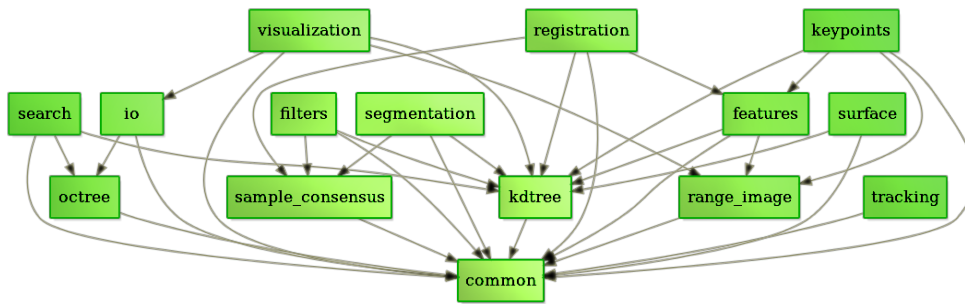


Figure 3.1: Point cloud library <sup>3</sup>

### 3.1.3 ROS

ROS <sup>4</sup> is a software framework for writing robot software. It was designed to ease the development of robot systems and to encourage a *collaborative* platform. The ROS ecosystem allows for a seamless integration between hardware drivers and software modules and aims to expedite the software prototyping, testing and deployment with several core libraries and developments tools available. ROS provides integration with other libraries that extend the core components, such as integration with OpenCV and PCL.

## 3.2 Camera Systems

Traditionally, there are two main techniques used in range sensing devices, triangulation and Time-of-Flight (ToF). Triangulation can be passive systems, like stereo vision, or active systems, such as structured light. Whereas, stereo vision systems analyse the divergence between two or more images taken at different positions to compute the object's range, structured light cameras project an infrared light pattern onto the scene and estimate the disparity given by the perspective distortion of the pattern. ToF cameras measure the time that light emitted by an illumination unit requires to travel to an object and back to a detector. The interest towards RGB-D based approaches increased widely when Microsoft released the first-generation Kinect device (Kinect V1) in 2010, for their XBox gaming console. Kinect being sold an order of magnitude cheaper than similar sensors that had existed at the time, reinvigorated interest in RGB-D sensors in the academic and industrial scenes. Functionally, there are several differences between structured light based RGB-D cameras and ToF based cameras.

### 3.2.1 Time of Flight System

Time of flight is a method for measuring the distance between a sensor and an object. Most ToF cameras use a Continuous Wave Intensity Modulation approach. The idea behind this technique is to actively illuminate the scene using a near infrared intensity-modulated, periodic light. A time shift  $\phi$  is caused in the optical signal of the sensor due to the time it takes for photons to travel the

<sup>4</sup><http://www.ros.org/>

distance between the camera and the object and back to the camera, at a finite speed (the speed of light  $c$ ). This optical shift is equivalent to a phase shift  $\Delta\varphi$  in the periodic signal. This shift is detected in each sensor pixel and can be easily transformed into the sensor-object distance, using  $c$  the speed of light and  $f$  the signal frequency,  $d = \frac{c\Delta\varphi}{4\pi f}$ . Figure 3.2 illustrates this principle.

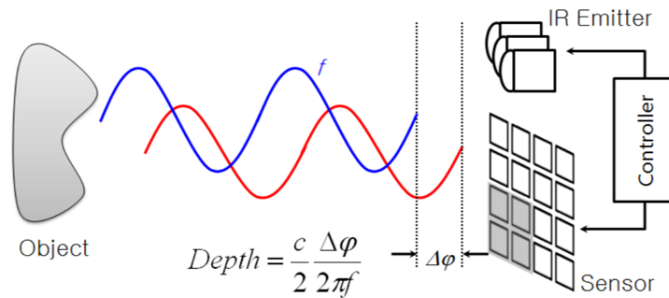


Figure 3.2: The principle of ToF depth camera: The phase delay between emitted and reflected IR signals are measured to calculate the distance from each sensor pixel to target objects [HLCH12]

ToF cameras are less sensitive to changes in the light conditions and are a more affordable technology when compared to structured light techniques. They also deliver a higher frame rate when compared to structured light cameras capturing a smoother geometry.

### 3.2.2 Structured Light System

The structured light technique is derived from stereo vision, but with one of the cameras being interchanged by a pattern projector. The source of light pattern can be a conventional 2D projector like the ones used for multimedia presentations. Both the sensors must be focused on the same scanning area and posed in well-known positions, determined by a system calibration. A sequence of known patterns is sequentially projected onto an object, which gets deformed by the geometric shape of the object. By analysing the distortion of the observed pattern, i.e. the disparity from the original projected pattern, depth information can be extracted. In this thesis, we will use the industrial sensor *Photoneo PhoXi® 3D Scanner S*, depicted in the figure 3.3 (the specifications can be seen in appendix B).



Figure 3.3: PhoXi 3D Scanner S

This sensor emits a set of coding patterns projected in succession, one after the other. These coding patterns encodes a spatial information. Using a succession of multiple patterns ensures a per pixel 3D information with a higher accuracy but it demands the scene to be static during the moment of capture. The camera captures the scanning area once per every projected pattern. For every “image point A”, the algorithm decodes the spatial information encoded in the succession of the intensity values captured by this point. This spatial information encodes the corresponding “image point B” in the view of the projector. By corresponding the two points an exact 3D position of the object point can be computed. Figure 3.4 depicts a graphical representation of the structured light principle.

PhoXi scanners are aimed at robot handling applications for bin picking, where randomly placed, semi-oriented objects can be picked and placed on a conveyor belt. These scanners are also compatible with multiple industrial robot brands.

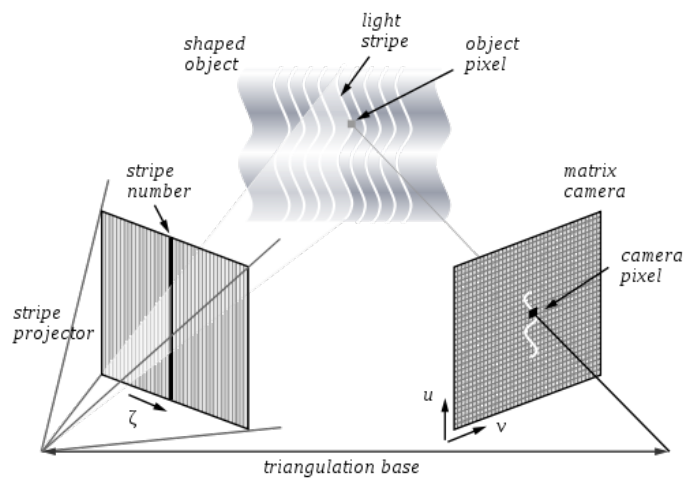


Figure 3.4: Structured Light Principle <sup>5</sup>

<sup>5</sup>[http://www.thefullwiki.org/Structured\\_Light\\_3D\\_Scanner](http://www.thefullwiki.org/Structured_Light_3D_Scanner)

## Chapter 4

# Recognition and 6 DoF Pose Estimation System

### 4.1 System Overview

The global recognition system was implemented as a ROS package and provides a 6DoF pose estimation for 3D objects. The package can receive sensor data through `sensor_msgs::PointCloud2`<sup>1</sup> messages and so is able to work directly with depth sensors. The system was designed to allow fast reconfigurations by using yaml files and the ROS parameter server. The system allows for a fast configuration of all the parameters of the methods used along the pipeline, since the choice of descriptor, to the configuration of the filters and classification distance.

### 4.2 Data Structures for Efficient Searches in Point Clouds

Most tasks that deal with point clouds are dependent on analysing the surroundings of a given point in the cloud, such as radius searches and  $K$  nearest neighbours. With the increasing sampling rates of spatial data points and the demand for precise and accurate results to be processed in real-time for a variety of tasks, efficient data structures are necessary to speed up the operations. The following sections will describe some space partition techniques.

#### 4.2.1 Octrees

An octree is a tree-based data structure for managing sparse 3-D data. It is a hierarchical space partition method that adapts the tree structure to the distribution of the points in the cloud. The root node describes a bounding box which encapsulates all points. Then, the algorithm recursively subdivides each voxel into eight disjoint octants with increasing level of resolution until the desired

---

<sup>1</sup>[http://docs.ros.org/api/sensor\\_msgs/html/msg/PointCloud2.html](http://docs.ros.org/api/sensor_msgs/html/msg/PointCloud2.html)

tree depth is reached or there are no more points present in that region of space. The result of this method can be seen in figure 4.1.

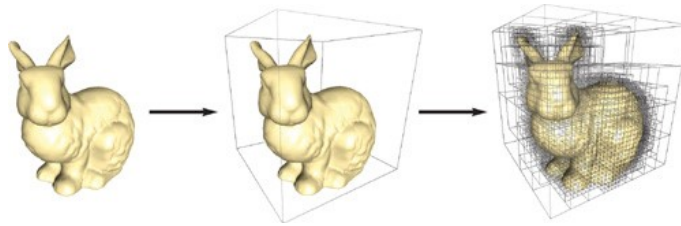


Figure 4.1: Octree of a bunny mesh model. <sup>2</sup>

#### 4.2.2 k-d trees

The k-d tree or a k-dimensional tree is a data structure used for organizing points in a space with k-dimensions. The k-d tree partitions a set of vectors along a k-dimensional hyper-plane, orthogonal to an axis and passing through the vector stored within the node. The dimension represented by the axis is known as the splitting dimension. Each node in the tree is defined by a plane through one of the dimensions that partitions the set of points into left/right (or up/down) sets, each with half the points of the parent node. These children are again partitioned into equal halves, using planes through a different dimension. The cheapest splitting algorithm simply looks at the parent of the element's splitting dimension,  $d$ , and splits the space along the median of points in each axis. This splitting algorithm is very effective on sets of points where each dimension is distributed using the same continuous distribution and parameters. Figure 4.2 illustrates the creation of a kd-tree in a 2D space.

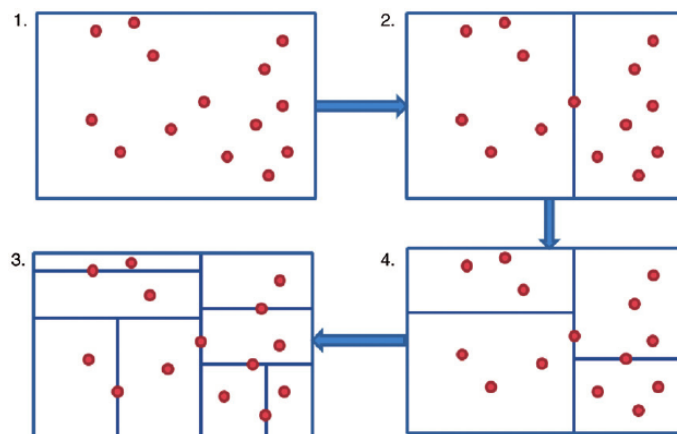


Figure 4.2: The four steps to create a 2D k-d tree. <sup>3</sup>

<sup>2</sup>[https://developer.nvidia.com/gpugems/GPUGems2/gpugems2\\_chapter37.html](https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter37.html)

<sup>3</sup>[https://www.researchgate.net/publication/254241712\\_Hierarchical\\_data\\_representation\\_structures\\_for\\_interactive\\_image\\_information\\_mining](https://www.researchgate.net/publication/254241712_Hierarchical_data_representation_structures_for_interactive_image_information_mining)

### 4.3 Normal Estimation

Though many different normal estimation techniques exist, the simplest method is based on the first order 3D plane fitting as proposed by [BC94]. The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation. Small portions a point cloud are fitted to temporary planes to generate the normal for a point. By using  $k$  number of neighbouring points from the point cloud, a covariance matrix is created, where the covariance matrix's eigenvectors and eigenvalues are the point of interest. After using Eigenvalue Decomposition (ED), the eigenvector associated with the smallest eigenvalue will be the normal vector for that point. The sign of eigenvectors from ED may be flipped, so to make sure that the normal calculated is oriented in the correct direction, the dot product of the reconstructed normal and a vector from the intersection point to the viewpoint origin is used.

### 4.4 Downsampling and Filtering

The time spent throughout the global recognition pipeline is proportional to the amount of points in the input scene point cloud and the size of the clouds representing the dataset models. Hence, using voxel grids to adapt the number of points in the point clouds provides some control in balancing the desired accuracy with the computational resources necessary. The accuracy of the system is also dependent on the quality of the input data. Filtering the data provided by the sensors mitigates the noise and outliers of the input clouds. Scanned point clouds are usually contaminated by numerous measurement outliers. Outliers are single points that are spread through the cloud. They are a measurement error due to the sensor's inaccuracy. These points are considered undesired noise, because they corrupt the results of the statistical analyses. Thus, removing the points from the cloud will not only make the computations faster, but also more precise. The implemented system allows the application of several customizable preprocessing algorithms (algorithm type and order of execution). The next section explains some of these methods.

#### 4.4.1 Voxel Grid Sampling

Using all the points in a point cloud can be time-consuming and especially nearby measurements may contain unnecessary details. For that reason, downsampling the cloud is an important preprocessing step. A voxel grid is a uniform space partition technique that can be used to cluster points according to their euclidean coordinates. The purpose of this filter is to downsample a given point cloud, maintaining the scene geometry but reducing the cloud size, i.e., the number of points. This filter creates a 3D voxel grid, like a set of small 3D boxes in space over the input point cloud data. Then, in each voxel all the points inside it will be replaced by a single point (i.e., downsampled). Typically, the entire voxel is replaced by the center or the centroid of the cluster. Whereas, selecting the center is less computationally heavy, the centroid represents the underlying surface more

accurately. Figure 4.3 shows the point cloud of a table before and after applying the voxel grid filter.

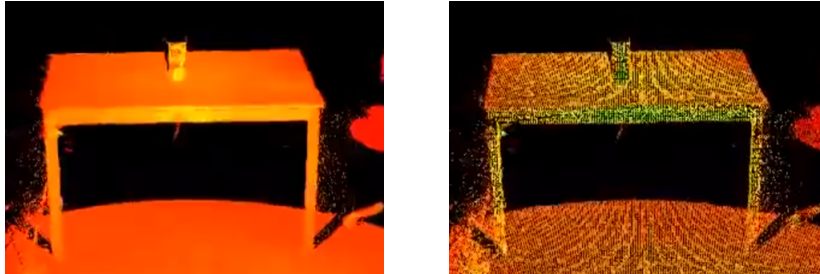


Figure 4.3: Point cloud of a table before (left) and after (right) voxel grid downsampling.<sup>4</sup>

#### 4.4.2 Passthrough Filter

The passthrough filter is a method used to select or remove points that fall outside the constraints of a particular field. For outlier removal, can be useful to single out needed objects from the cloud by selecting points that lie inside a given bounding box (useful if the study environment area is known *a priori*) or remove points outside color or intensity constraints.

#### 4.4.3 Radius Outlier Removal Filter

The radius-based outlier removal filter is the simplest method to remove outliers. A search radius  $r$  must be specified along with the minimum number of neighbours  $K$  that a point should have to not be labelled and outlier. The algorithm will then iterate through all of the points in the cloud (which can be extremely slow if the cloud is big) and check the number of neighbours found within the specified radius. If it is less than the desired, the point is removed. It is specially useful for when the density of the point cloud is known and very effective to remove isolated points.

#### 4.4.4 Statistical Outlier Removal Filter

The statistical outlier removal filter removes points by calculating for all points the mean distance to the nearest points. The algorithm assumes all points follow a Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$ . If a point has a mean distance which is greater than a given threshold, usually some percentage of  $\sigma$ , they are treated as outliers and removed. Figure 4.4 shows a point cloud of a table before and after applying the statistical outlier removal filter.

<sup>4</sup>[http://pointclouds.org/documentation/tutorials/voxel\\_grid.php/](http://pointclouds.org/documentation/tutorials/voxel_grid.php/)

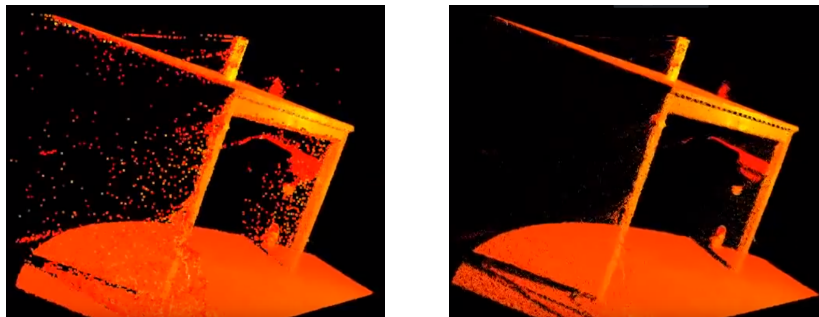


Figure 4.4: Point cloud of a table before (left) and after (right) statistical outlier removal.<sup>5</sup>

## 4.5 Training Phase

To train the system, the CAD meshes representing the objects of the datasets are transformed into partial point clouds, simulating the input that will be given by a depth sensor. A virtual camera is placed around the object on a bounding sphere with a radius large enough to enclose the object. The partial views are generated placing the camera at the center of the triangular faces of a regular icosahedron which encloses the model under consideration. For each generated partial view of a model, the desired 3D features are computed and stored for future use in the matching step. Alongside the views, a transformation between the model and the partial view coordinates is also stored. The inverse transformation allows the conversion of the partial view to model coordinates and, therefore, by fusing all partial views, the full 3D model can be obtained. Figure 4.5 shows an example of pointclouds generated from a partial view of the *Stanford bunny* mesh model.

## 4.6 Testing Phase

The next few sections will describe the different stages comprising the testing phase of the global pipeline. An overview of recognition pipeline can be seen in the figure 4.6.

### 4.6.1 Segmentation

Global features require the notion of object, hence the first step in the recognition pipeline involves the segmentation of the input scene to extract the objects present. First, is necessary to identify and isolate the area of interest. For that purpose, is possible to filter the cloud with a passthrough filter (detailed in section 4.4) and extract points within the ROI or use a method based on the extraction of a dominant scene plane in which the objects are laid on top. The final step consists in a euclidean clustering algorithm to isolate each object. A possible output from this approach can be seen in figure 4.7.

<sup>5</sup>[http://www.pointclouds.org/documentation/tutorials/statistical\\_outlier.php](http://www.pointclouds.org/documentation/tutorials/statistical_outlier.php)

<sup>6</sup><http://graphics.stanford.edu/data/3Dscanrep/>

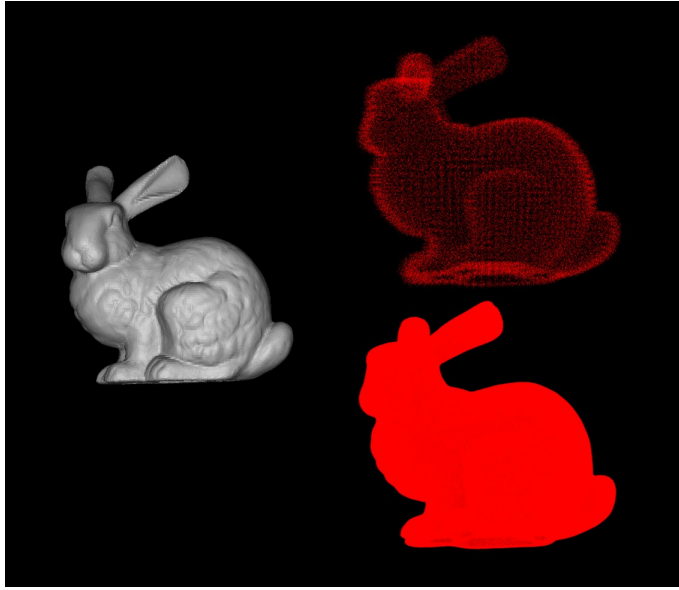


Figure 4.5: The *Stanford bunny* mesh model and two generated views from the same viewpoint but different resolutions. <sup>6</sup>

#### 4.6.1.1 Extract Dominant Plane

Plane segmentation consists in identifying the horizontal plane on which the objects lie and extracting this plane from the point cloud in order to facilitate the clustering of the remaining objects. In order to extract the plane from the scene the Random Sample Consensus (RANSAC) is used. If we denote by  $P$  the scene point cloud, a plane  $H$  can be described using four parameters  $\{a, b, c, d\}$  and is defined as follows:

$$H = \{(x, y, z)^T \in P | ax + by + cz + d = 0\} \quad (4.1)$$

To estimate the parameters  $\{a, b, c, d\}$ , the RANSAC algorithm first selects three random points from the point cloud and calculates the four parameters of the plane  $P_1$  defined by the three points. Then, the algorithm computes the euclidean distances of all points  $\mathbf{p} \in P$  to the plane  $P_1$ . A point  $\mathbf{p}$  of  $P$  is considered to be a point of  $P_1$  if its distance  $d$  to  $P_1$  satisfies the threshold provided as the minimum distance. The algorithm then estimates the total number of points belonging to  $P_1$ . This process repeats  $n$  times and finally the plane model  $P_i \in [1, n[$  that contains the maximum number of points is considered the dominant plane and extracted from the scene.

#### 4.6.1.2 Euclidean Clustering

Euclidean clustering groups points that are close together. The clustering method is guided by a neighbouring threshold  $t$ , which indicates how close two points are required to be to belong to the same object. Therefore, for a successful segmentation, the method requires different objects to be at least  $t$  far away from each other. The algorithm starts a cluster with a random point in the scene

## Recognition and 6 DoF Pose Estimation System

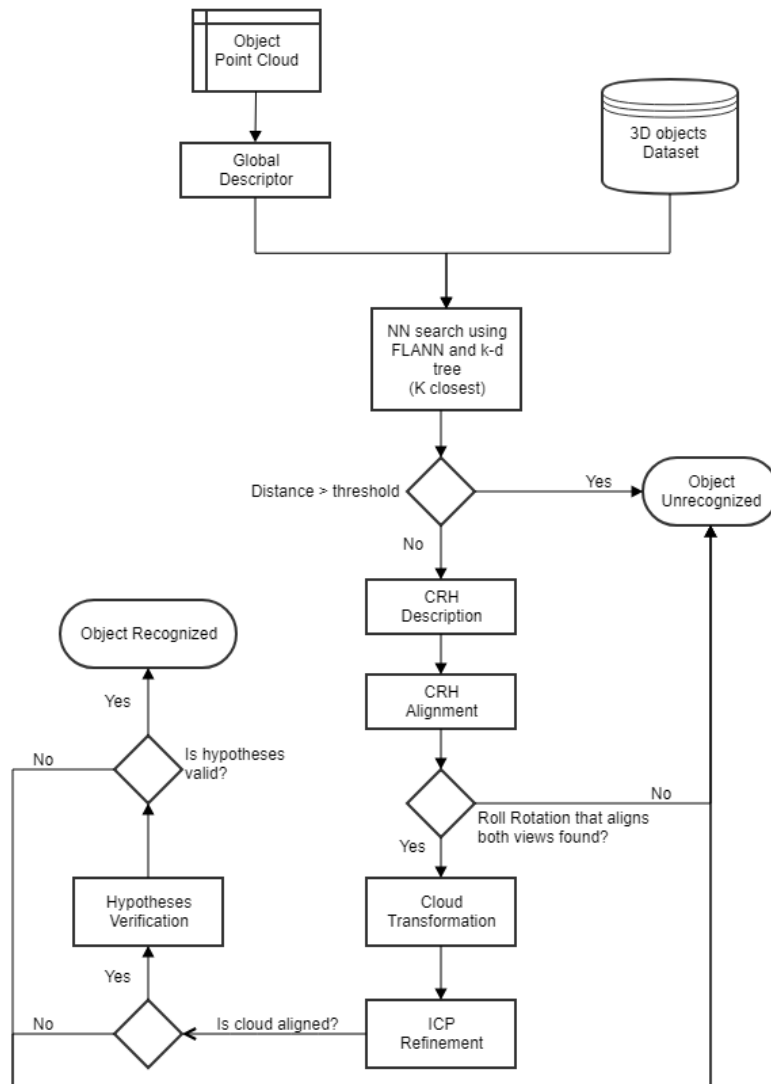


Figure 4.6: Simplified 3D object recognition pipeline

point cloud. All neighbouring points within a distance threshold from the point are added to the cluster. This then continues for the points in the cluster until no more points can be added. At that point a new cluster is started and the above is repeated. All clusters sufficiently large are kept. Another useful feature is that it is possible to set a minimum and a maximum number of points that a cluster can contain. This helps to filter out noise (isolated single points) or parts of the surface that weren't segmented out. A K-d tree structure for finding the nearest neighbours was used to speed up the computations.

### 4.6.2 Features Description and Matching

The clusters computed in the segmentation phase represent the candidate objects in the scene. The geometry and shape of each object is described by the desired global descriptor and represented

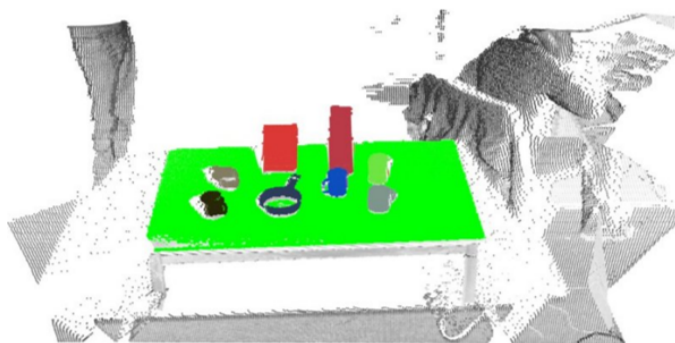


Figure 4.7: A segmentation of a table scene and the object clusters found to lie on it from an unorganized point cloud dataset. [RBMB09]

by a single histogram (except for CVFH where multiple histograms might be obtained). The histogram(s) are independently compared against all the ones in the database (obtained during the training stage) and the  $K$  models NNs are retrieved. The  $K$  neighbours represent the most similar views in the training set based on the description obtained by means of the desired global feature and the metric used to compare the histograms (L1, L2 or L2-Simple). Usually, retrieving more than a single  $K$  neighbour is advised, as the results can be greatly improved through pose estimation and postprocessing. Notwithstanding, the more descriptive and distinctive a feature is, the better the matching algorithm can prune unfeasible candidates. The histogram matching is done with a k-d tree (implemented in the FLANN library)

### 4.6.3 Camera Roll and 6-DoF Pose

The partial view candidates obtained by matching an object cluster to the training set can be aligned using their respective centroids obtaining a 5DoF pose. The 3DoF translation is computed from the differences of the centroids. By aligning the average normals of the two clusters, a 2DoF rotation is estimated, but this leaves the last rotation along the camera axis unknown. Global features are usually invariant to this rotation since rotations about the roll axis of the camera do not result in changes of the geometry of the visible part of the object. Even though the camera roll descriptor is not able to describe the shape of the object, is not invariant to along the angle of the camera roll, hence can be used to complement the global descriptors. To compute de camera roll descriptor, each point's normal is projected onto the plane orthogonal to the vector given by the camera center and the centroid of the given cluster. The histogram is binned by the angles of the projected normals relative to the camera's up-view vector. By matching the two histograms, the full 6DoF pose can be estimated.

### 4.6.4 Post Processing

To improve the outcome of the recognition, it is possible to add a final postprocessing stage. A first step consists in applying the iterative closest point (ICP) to the recognition hypotheses, to refine

## Recognition and 6 DoF Pose Estimation System

the estimated 6DoF pose. Next, it is also possible to apply a hypotheses verification, that aims to reduce the number of false positives while maintaining correct recognitions. This step usually relies on geometric cues that are usually computed after the models have been aligned.

## Chapter 5

# Experiments and Results

This section will describe the experiments done to test the global pipeline and the results obtained.

### 5.1 The Dataset

The dataset to test the global pipeline consists on a set of models made available from the use cases of Simoldes and PSA of the Scalable4.0 project. For the dataset used in our experiments we used 5 models of those provided. The models are named as such: *tray*, *filter*, *cap*, *8080* and *8537*. For each model, there is also available a CAD model. Figures of the dataset can be seen in appendix [A](#).

Each object was passed through the training stage to generate artificial views of the object and to compute the different feature descriptors. In our experiments we used the VFH, ESF and CVFH descriptors provided by PCL. Others descriptors were implemented into the pipeline but will not be explored in this section. Each model generated 161 views so the final dataset is comprised with 805 models. The fully fused point clouds obtained from the CAD models during the training stage can be seen in following figures, figure [5.1](#) to figure [5.5](#):



Figure 5.1: Two views from the full generated *tray* pointcloud.



Figure 5.2: Two views from the full generated *filter* pointcloud.



Figure 5.3: Two views from the full generated *cap* pointcloud.



Figure 5.4: Two views from the full generated *8080* pointcloud.



Figure 5.5: Two views from the full generated *8537* pointcloud.

## 5.2 Experiments

To validate the dataset we used a sample scene with all the objects present. The objects are arranged in an orderly manner as the system is to be validated in scenarios under a controlled environment. The test scene is depicted in figure 5.6.

The point clouds were collected using a structured light sensor, the *PhoXi®3D Scanner S* (specification in appendix B). The same scene was scanned with different camera tilt angles ranging from  $15^\circ$  and  $-45^\circ$ . The figure 5.7 displays on the left the acquired point cloud after been downsampled using the filter voxel grid with a leaf of  $1mm$  and after using a statistical outlier

## Experiments and Results



Figure 5.6: Picture of the test scene used in the experiments.

removal to remove noise. On the right, the dominant plane was extracted using the RANSAC algorithm. Some of the points belonging to the objects were removed due to being considered part of the plane (if they are too close). This can be mitigated or avoided if a passthrough filter is used to select the region of interest, in the situations where this region is known *a priori*.

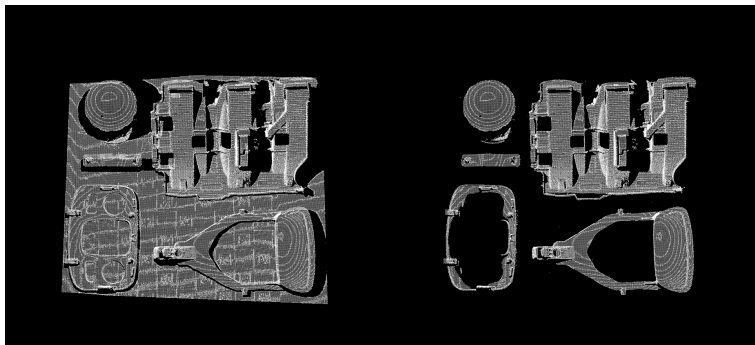


Figure 5.7: Point cloud of the test scene after the preprocessing step (left) and after the extraction of the dominant plane (right).

Due to the nature of the scene, the clustering of the pointcloud into clusters of candidate objects produced good results as seen in the figure 5.8

In order to validate the recognition matching step, we choose confusion matrices because they are useful for visualizing the performance of a classification algorithm. Each of the scenes (with different angles) contain the 5 objects we need to identify. Each cluster representing a candidate object is matched with the 805 models in the dataset. The nearest neighbour is matched with the object in the scene.

The ESF descriptor was capable of recognizing the correct object for all clusters. The performance of the ESF descriptor on a specific class varies depending on the presence of similarly shaped classes, with similar partial views also being able to introduce some confusion, a problem not very challenging for our dataset.

The VFH was capable of recognizing 4 of the 5 objects in the dataset. The *cap* model, in

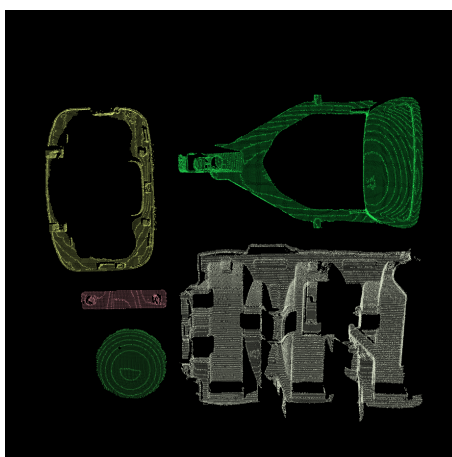


Figure 5.8: The computed clusters by the euclidean clustering algorithm.

two instances was classified as another object. For higher tilt angles, the occlusions in the scene increase and the VFH descriptor is not able to handle large missing parts from the object.

The CFVH descriptor was not capable of correctly classify the objects. Apart from the *filter* with 3 correct classifications and the *8537* with 2 correct classifications in 5, this descriptor was not able to classify correctly even once one of the other objects. The CFVH descriptor was created to be able to handle missing parts by robustly estimating patches (smaller surfaces) using information from the normals and position as a reference system to describe the rest of the object from that viewpoint. However, the various stable patches are estimated both in training and recognition. If those patches cannot be found in the cluster to be classified, the C VFH will not be able to recognized the objects. Also, the absence of the notion of an aligned euclidean space causes the C VFH descriptor to lose the capability for proper spatial description.

The tables 5.1 to 5.3 display the confusion matrices for the classifications in the various scenes for each descriptor.

Table 5.1: The confusion matrix for the clusters classification using VFH descriptors

	Cap	Filter	Tray	8080	8537
Cap	2	0	0	1	2
Filter	0	5	0	0	0
Tray	0	0	5	0	0
8080	0	0	0	5	0
8537	0	0	0	0	5

Using  $K$  nearest neighbours from the feature matching step, the system follows up with the pose estimation. For the pose estimation, we complement the global descriptor with the camera roll descriptor for a full 6DoF pose estimation. The final poses are the result of a pose refinement using the ICP algorithm. A hypotheses verification is done in all the recognition hypotheses to eliminate false positives. The estimated pose that minimises the Mean Square Error (MSE) is retrieved. The MSE errors are calculated using the distances of the correspondence points computed by the

## Experiments and Results

Table 5.2: The confusion matrix for the clusters classification using ESF descriptors

	Cap	Filter	Tray	8080	8537
Cap	5	0	0	0	0
Filter	0	5	0	0	0
Tray	0	0	5	0	0
8080	0	0	0	5	0
8537	0	0	0	0	5

Table 5.3: The confusion matrix for the clusters classification using CVFH descriptors

	cap	filter	tray	8080	8537
cap	0	0	3	1	1
filter	0	3	0	0	2
tray	0	2	0	2	1
8080	2	0	1	0	2
8537	0	0	3	0	2

ICP algorithm. The ICP algorithm determines the rigid transformation between the downsampled pointcloud of the different views of the dataset model chosen during the matching phase and the pointclouds of each cluster, after the captured pointcloud has been filtered and downsampled. Table 5.4 displays the number of points of the pointclouds for each cluster for the different camera tilt angles and the average number of points for the views of the dataset model with a variation of a standard deviation.

Table 5.4: Number of points of the dataset models views and the clusters for determining the rigid transformation between the using the ICP algorithm.

	Number of Points					
	Dataset View Model	-45	-30	-15	0	15
Tray	$1397 \pm 435$	1933	2109	2405	2498	2779
Filter	$247 \pm 51$	544	294	257	245	585
Cap	$247 \pm 51$	141	128	123	114	106
8080	$564 \pm 175$	866	927	924	825	848
8537	$509 \pm 130$	497	582	563	494	427

The MSE errors are displayed in the tables 5.5 to 5.9, with an error of one standard deviation. Due to the poor accuracy of the CVFH descriptor in classify the clusters, this analysis was not viable.

From the analysis of the tables, the 8080 model was the one with an overall pose estimation with the minimum MSE. However, for the scene with an angle of  $-45$  degrees the pipeline was not able to estimate the pose for this model. Besides the missing data in the *cap* distances for the VFH descriptor, due to not being correctly classified, the pipeline was not able to estimate the pose of the model for matched view of the dataset for the two scenes it was recognized. This poor pose estimations happen when the camera roll descriptor is not able to correctly estimate the last

## Experiments and Results

Table 5.5: MSE for the *tray* pose estimations.

Angle	MSE / $10^{-5}m$ (VFH)	MSE / $10^{-5}m$ (ESF)
-45	$1.232 \pm 0.045$	$1.016 \pm 0.044$
-30	$1.015 \pm 0.039$	$0.944 \pm 0.041$
-15	$0.746 \pm 0.029$	$0.671 \pm 0.028$
0	$0.684 \pm 0.029$	$0.500 \pm 0.022$
15	$0.639 \pm 0.029$	$0.579 \pm 0.024$

Table 5.6: MSE for the *filter* pose estimations.

Angle	MSE / $10^{-5}m$ (VFH)	MSE / $10^{-5}m$ (ESF)
-45	$1.010 \pm 0.065$	$0.938 \pm 0.066$
-30	$0.323 \pm 0.025$	$0.297 \pm 0.019$
-15	$0.219 \pm 0.018$	$0.211 \pm 0.020$
0	$0.232 \pm 0.025$	$0.195 \pm 0.021$
15	$1.615 \pm 0.067$	$1.068 \pm 0.057$

Table 5.7: MSE for the *cap* pose estimations.

Angle	MSE / $10^{-5}m$ (VFH)	MSE / $10^{-5}m$ (ESF)
-45	-	$0.909 \pm 0.117$
-30	-	$0.875 \pm 0.139$
-15	-	$0.733 \pm 0.120$
0	-	$1.225 \pm 0.141$
15	-	$1.450 \pm 0.198$

Table 5.8: MSE for the *8080* pose estimations.

Angle	MSE / $10^{-5}m$ (VFH)	MSE / $10^{-5}m$ (ESF)
-45	$0.520 \pm 0.039$	-
-30	$0.570 \pm 0.038$	$0.654 \pm 0.040$
-15	$0.491 \pm 0.028$	$0.500 \pm 0.026$
0	$0.729 \pm 0.038$	$0.491 \pm 0.028$
15	$0.529 \pm 0.032$	$0.509 \pm 0.034$

rotation along the camera axis. The ICP algorithm requires a good initial guess of the rigid body transformation to be able to align two point clouds.

The MSE values are generally higher for the VFH descriptor when compared to the ESF descriptors. Also, to obtain these results the VFH descriptor required 10 nearest neighbours from the matched features, whereas the ESF descriptor only required 5, increasing the processing time. Typical requirements from Scalable4.0 use-cases set maximum error of  $1cm$ . Our pipeline estimated the pose of the objects with an error no greater than  $2 * 10^{-5}m$  complying with their

## Experiments and Results

Table 5.9: MSE for the 8537 pose estimations.

Angle	MSE / $10^{-5}m$ (VFH)	MSE / $10^{-5}m$ (ESF)
-45	$1.145 \pm 0.080$	$1.632 \pm 0.105$
-30	$0.971 \pm 0.071$	$1.307 \pm 0.084$
-15	$1.077 \pm 0.073$	$0.784 \pm 0.069$
0	$1.430 \pm 0.100$	$0.774 \pm 0.066$
15	$1.025 \pm 0.088$	$1.521 \pm 0.098$

requirements.

The figure 5.9 shows the pose estimation for the 5 objects in the dataset with the matched clusters in the scene. The points in green represent the scene and the points in blue the points of the matched view.

Experiments and Results

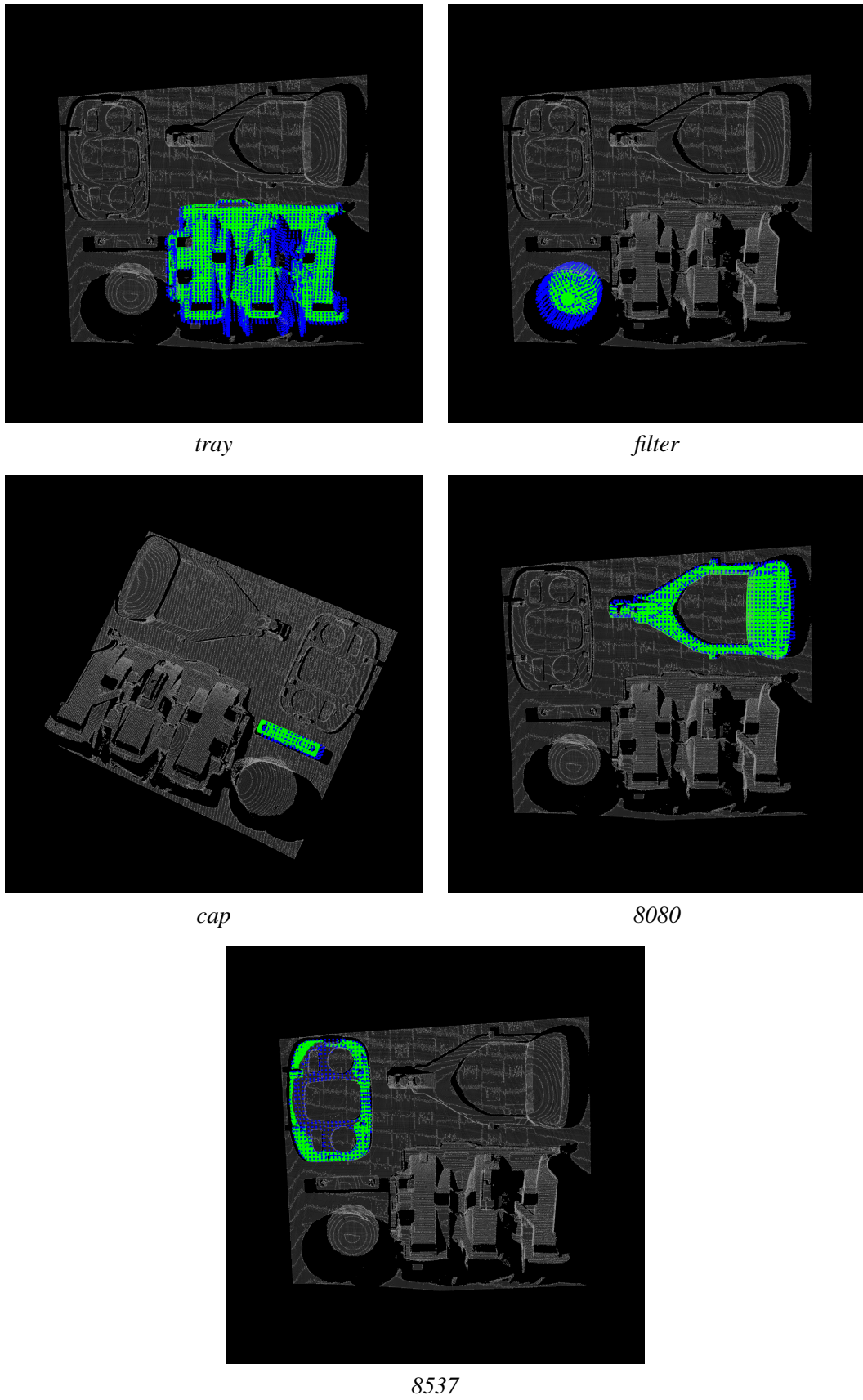


Figure 5.9: Pose estimations using ESF descriptors.

## Chapter 6

# Conclusions and Future Work

This dissertation has implemented and evaluated a system for object recognition and 6DoF pose estimation of 3D models in depth sensor data. The pipeline was tested with a dataset consisted of 5 objects, a subset of the objects provided by the PSA and Simoldes use cases from the project Scalable4.0. The experimental setup consisted in a single scene captured with different camera tilt angles and in controlled conditions.

The experiments showed that for controlled environments with little to none clutter, the pipeline was able to segment the scene in clusters with the correct object candidates. Whereas the VFH and ESF descriptors were able to recognize the objects (apart from the *cap* in the case of the VFH) with a good accuracy, the CVFH proved incapable to provide overall good results, showing the correct choice of descriptor for the scene is important for the robustness of the system.

For the pose estimation the ICP worked satisfactory as pose refinement method and the hypotheses verification was able to remove false positives. The final pose estimations had an MSE no greater than  $2 * 10^{-5}m$  complying with the accuracy requirements of the project. Typical requirements from Scalable4.0 use-cases set maximum error of  $1cm$  for a robotic arm be able to grasp objects, so the errors obtained during our experiments prove that our system is a feasible application to grasp the dataset objects within the Scalable4.0 environment.

In some situations, the nearest neighbours' views retrieved when matching the features, didn't allow for the camera roll descriptor to compute the correct angle along camera axis, preventing from obtaining a pose estimation.

This thesis successfully implemented a customizable ROS package for a 3D Global Recognition and 6DoF Pose Estimation pipeline. From the *yaml* files, it's possible to customize the pipeline to the needs of the task in hand. The end user can choose the filters to be applied to the pointcloud and in which order; can use the global descriptor better suited to their problem; the criteria to match the descriptors; if the post processing step is necessary or not for the desired accuracy. Although the focus was put on using dataset with 5 objects already processed the system is fully capable to directly receive sensor data.

## Conclusions and Future Work

For future work, some additional research could be done in terms of the classification algorithms to minimize the number of incorrectly classified objects. Another interesting topic would be to study the pipeline with clutter scenes and implement the necessary features into the system to deal with this problem. Finally, realtime or close to realtime 3D object recognition is possible with more optimized code and multicore processing.

# References

- [AMT<sup>+</sup>12] A. Aldoma, Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics Automation Magazine*, 19(3):80–91, Sept 2012.
- [AVB<sup>+</sup>11] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592, Nov 2011.
- [BC94] J. Berkmann and T. Caelli. Computation of surface geometry and segmentation using covariance techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1114–1116, Nov 1994.
- [BM92] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.
- [FA14] S. Filipe and L. A. Alexandre. A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset. In *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*, volume 1, pages 476–483, Jan 2014.
- [FHK<sup>+</sup>04] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *Computer Vision - ECCV 2004*, pages 224–237, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [HHRB12] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Robot soccer world cup xv. chapter Real-time Plane Segmentation Using RGB-D Cameras, pages 306–317. Springer-Verlag, Berlin, Heidelberg, 2012.
- [HLCH12] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Publishing Company, Incorporated, 2012.
- [HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.
- [JH98] A.E. Johnson and M. Hebert. Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16(9):635 – 651, 1998.
- [Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

## REFERENCES

- [RBB09] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009.
- [RBMB08] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, Sept 2008.
- [RBMB09] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6, Oct 2009.
- [RBTH10] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, Oct 2010.
- [RC11] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.
- [Rus13] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Robot Manipulation*, volume 85. Springer Tracts in Advanced Robotics, 2013.
- [SB97] Stephen M. Smith and J. Michael Brady. Susan—a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, May 1997.
- [SRS12] Wandu Susanto, Marcus Rohrbach, and Bernt Schiele. 3d object detection with multiple kinects. In *Proceedings of the 12th International Conference on Computer Vision - Volume 2, ECCV’12*, pages 93–102, Berlin, Heidelberg, 2012. Springer-Verlag.
- [TSDS10] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 356–369, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [WV11] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2987–2992, Dec 2011.
- [Zho09] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696, 2009.

## Appendix A

### The Dataset

Dataset used to train and test the global pipeline. It consists of five models provided by the PSA and Simoldes use cases of the Scalable4.0 project.



Figure A.1: Two views from the *tray*.

## The Dataset



Figure A.2: Two views from the *filter* object.



Figure A.3: Two views from the *cap* object.

## The Dataset

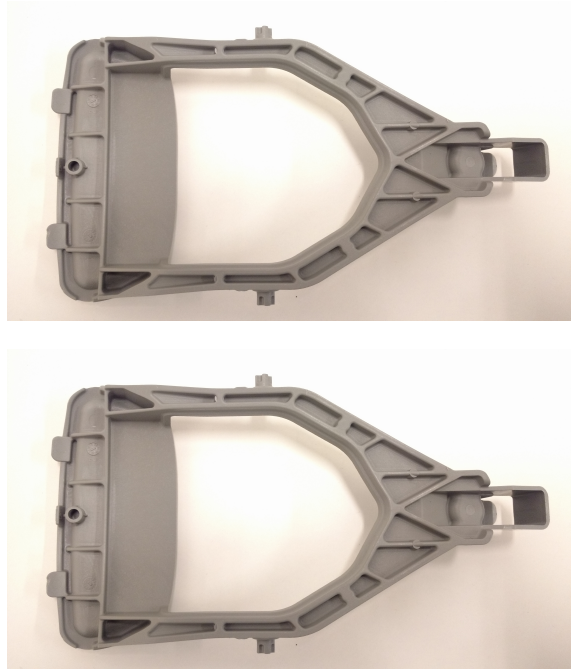


Figure A.4: Two views from the 8080 object.



Figure A.5: Two views from the 8537 object.

## Appendix B

# Photoneo PhoXi® 3D Scanner S

The specification of the sensor used in the experiments.

Download: [PhoXi 3D Scanner - User Manual](#)

Parameter	Values	
	High res mode	High speed
Mode	High res mode	High speed
Depth Map resolution (independent measurements)	3.2M points	0.8M points
Point size	0.16 mm	0.32 mm
Absolute accuracy	≤ 50 μm	≤ 50 μm
Z noise	≤ 50 μm	≤ 50 μm
FPS	5 fps	6.6 fps
Data acquisition time – best case: white diffuse objects	200 ms	150 ms
Data acquisition time – worst case: dark objects	2 s	1 s
3D points throughput	16 Million points per second	
GPU	NVIDIA Maxwell™ 1 TFLOPS with 256 NVIDIA® CUDA® Cores	
Scanner Dimensions	77 x 68 x 296 mm	
Baseline	230 mm	
Minimal recommended distance	384 mm	
Maximal recommended distance	520 mm	
Weight	900 g	

