



UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



Technische Universität München

3D RECONSTRUCTION AND RECOGNITION OF OBJECTS USING A KINECT CAMERA

Bachelor Thesis

Submitted to the Faculty of Electrical Engineering

Technische Universität München

In fulfilment of the requirements for the degree in

Audiovisual Systems Engineering

Universitat Politècnica de Catalunya

by

Alba Luján Rossell

Advisor: Dr. Claus Lenz

Supervisor: Josep Ramon Casas

Munich, September 2016

Abstract

This project is based on the recent advances of 3D depth cameras, one being the Microsoft Kinect sensors. Nowadays, this technology enables to create really accurate 3D figures and scenes simplifying difficult tasks in Robotics and embedded systems.

The purpose of this project consists of using both the color data and this depth sensing technology to reconstruct and recognize objects in a scene. The data from the Kinect sensor is processed first to eliminate the planes of the scene to get a better perception of the elements we may find, and consequently, to obtain relevant features of every single object for the creation of a model for every different class.

The results show the performance of the system, by testing a database of different elements against the training of some specific objects previously selected: Bowls, pillows and monitors. We can conclude that the classification using the color information is more accurate than using 3D data, even though in both cases the results are quite satisfactory.

Resum

Aquest projecte es basa en els avenços recents en càmeres de profunditat 3D, sent un dels sensors, el Kinect de Microsoft. Avui en dia, aquesta tecnologia permet crear figures i escenes en 3D molt més precises, fet que simplifica complexes tasques en Robòtica i Embedded Systems.

L'objectiu d'aquest projecte, consisteix tant en l'ús tant de les dades de color com de profunditat que proporciona el sensor del Kinect, per reconstruir i reconèixer objectes en una escena. En primer lloc, es processa aquesta informació per eliminar els plans de l'escena corresponents a parets o grans superfícies per obtenir una millor percepció dels elements que podem trobar en la imatge i, en conseqüència, obtenir característiques rellevants de cada objecte, per a la creació d'un model per a cada classe diferent .

Els resultats mostren el rendiment del sistema, testejant una base de dades de diferents elements, en contra l'entrenament previ d'alguns objectes específics que hem seleccionat: bols, coixins i pantalles. Podem concloure que la classificació utilitzant la informació de color és més precisa que amb l'ús de les dades 3D. Tot així, en ambdós casos, el resultat és suficientment satisfactori.

Resumen

Este proyecto se basa en los más recientes avances en cámaras de profundidad 3D, siendo uno de los sensores, el Kinect de Microsoft. Hoy en día, esta tecnología permite crear figuras y escenas 3D mucho más precisas, lo que simplifica complejas tareas en Robótica y Embedded Systems.

El objetivo de este proyecto, consiste tanto en el uso tanto de los datos de color como de profundidad que proporciona el sensor del Kinect, para reconstruir y reconocer objetos en una escena. En primer lugar, se procesa esta información para eliminar los planos de la escena correspondientes a paredes o grandes superficies para obtener una mejor percepción de los elementos que podemos encontrar en la imagen y, en consecuencia, obtener características relevantes de cada objeto, para la creación de un modelo de cada clase diferente.

Los resultados muestran el rendimiento del sistema, testeando una base de datos de diferentes elementos, en oposición al entrenamiento previo de algunos objetos específicos que hemos seleccionado: cuencos, almohadas y pantallas. Podemos concluir que la clasificación utilizando la información de color es más precisa que con el uso de los datos 3D. Aún así, en ambos casos el resultado es suficientemente satisfactorio.



Acknowledgements

I would like to express my gratitude to my supervisor, Claus Lenz, at TUM for the advice given and all the support provided during the development of this thesis. Also, I would like to thank professor Josep Ram3n Casas for the help and supervision from Spain, and professor Philipp Tiefenbacher for making possible the realization of this Bachelor thesis in the Electrical faculty of TUM.

Finally, I would like to thank the Polytechnic University of Catalonia and the Technical University of Munich, for giving me the opportunity to realize this project abroad and special thanks to my family for the support always received.

Revision history and approval record

Revision	Date	Purpose
0	07/09/2016	Document creation
1	10/09/2016	Document revision
2	19/09/2016	Document revision
3	27/09/2016	Final revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Alba Luján Rossell	alba.lujan@alu-etsetb.upc.edu
Project Supervisor TUM: Claus Lenz	lenz@in.tum.de
Project Supervisor 2 TUM: F	philipp.tiefenbacher@tum.de
Project Supervisor UPC: Josep Ramon Casas	josep.ramon.casas@upc.edu

Written by:		Reviewed and approved by:	
Date	07/09/2016	Date	27/09/2016
Name	Alba Luján	Name	Claus Lenz
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract	1
Resum	2
Resumen	3
Acknowledgements	4
Revision history and approval record	5
Table of contents	6
List of Figures	8
List of Tables	9
1. Introduction	10
1.1. Statement of purpose	10
1.2. Motivation	10
1.3. Requirements and specification	10
1.4. Methods and procedures	11
1.5. Work Plan	11
1.6. Milestones	14
1.7. Gantt Diagram	14
1.8. Deviations and incidences	15
2. State of the art of the technology used or applied in this thesis:	16
2.1. Kinect sensor and Depth Estimation	16
2.2. Databases	17
2.3. Object Recognition and Design of own approach	18
2.4. Point Cloud Data	18
2.5. Fast Point Feature Histogram	21
3. Methodology / project development:	24
3.1. Features Extraction	24
3.1.1 Colour Segmentation	26
3.1.2 Point Cloud Creation	27
3.1.3 Plane Detection	28
3.1.4 Object Reconstruction	29
3.2. Training and Testing	31
3.2.1 Test Database	32
3.2.2 Test Algorithms	32
3.2.2.1 Testing a model	32

3.2.2.2 Testing more than one model	32
3.2.3 Evaluation of the classifier	33
4. Results	34
4.1. Results obtained in the features extraction	34
4.1.1 Colour Segmentation	34
4.1.1.1 Opening method	34
4.1.1.2 K-means Clustering	35
4.1.1.3 Results of the final segmentation	36
4.1.2 Point Cloud Creation.....	37
4.1.3 Plane Detection	38
4.1.4 Object Reconstruction	40
4.2. Testing results	41
4.2.1 Results testing one model.....	41
4.2.2 Results testing more than one model.....	45
5. Budget.....	48
6. Conclusions and future development:.....	49
Bibliography:.....	50
Glossary	51

List of Figures

Figure 1. Gantt Diagram	14
Figure 2. Scheme of a Microsoft Kinect Sensor	16
Figure 3. Trigonometry for depth calculation	16
Figure 4. Number of representations for every object in the DB	17
Figure 5. Scheme for object recognition.....	18
Figure 6. Ideal pinhole camera model	19
Figure 7. Model of PFH for a pair of points	21
Figure 8. Model of a region diagram PFH	22
Figure 9. Model of a region diagram FPFH	23
Figure 10. General Scheme for feature extraction	25
Figure 11. Example of Colour Segmentation	27
Figure 12. Example of Point Cloud	27
Figure 13. Scheme of Plane Detection	28
Figure 14. Example of Plane Detection.....	28
Figure 15. Example of Remain Point Cloud	29
Figure 16. Example of Object Reconstruction	30
Figure 17. General scheme for testing	31
Figure 18. Scheme for the evaluation of a classifier	33
Figure 19. Examples of segmentations using different elements and shapes	34
Figure 20. Examples of segmentation using a different number of Clusters and Loops ..	35
Figure 21. Examples of the applied binary mask.....	36
Figure 22. Results for the Point Cloud Creation	37
Figure 23. Results for the Plane Detection.....	38
Figure 24. Results for the Object Reconstruction.....	40
Figure 25. Results for test 1	42
Figure 26. Combination of Color and Depth Features	43
Figure 27. Performance of the baseline detector described in another project	44
Figure 28. Results for test 2.....	45
Figure 29. Results for test 2 using different weights.....	46
Figure 30. Comparing Precision/Recall curves	46

List of Tables:

Table 1. Work Package 1	11
Table 2. Work Package 2	11
Table 3. Work Package 3	12
Table 4. Work Package 4	12
Table 5. Work Package 5	12
Table 6. Work Package 6	12
Table 7. Work Package 7	13
Table 8. Work Package 8	13
Table 9. Work Package 9	13
Table 10. Work Package 10.....	13
Table 11. Milestones.....	14
Table 12. Intrinsic Parameters of Depth Camera(Kinect v1)	27
Table 13. Budget of the project.....	48

1. Introduction

1.1. Statement of purpose

The main objective of this thesis is to implement an algorithm based on machine learning capable of both identify and recognize objects in a scene, by extracting the color information of them as well as their representation in 3D. The database of color and depth images is provided by a Kinect.

For this purpose, one of the first goals consists on an intensive exploration of the state of art in the fields of 3D object reconstruction and recognition, and an evaluation and benchmarking of the existing approaches.

Our own approach is developed and implemented in MATLAB, in order to compare and decide which features can contribute with more valuable information, as well as determine which steps need to be done to achieve a good quality classifier.

1.2. Motivation

The general motivation for object recognition, relies on the latest improvement of depth cameras technology. The recent progress on this matter, aroused the interest of researchers and enthusiasts of robotics and embedded systems. Object recognition, can become indispensable in tasks for human-robot interaction, as well as really helpful in a medical environments, by simplifying difficult tasks. Nowadays, it is also used very often for augmented reality applications.

1.3. Requirements and specifications

The requirements to conduct this project are, on the one hand, technical skills such as mathematical background knowledge, insight in computer vision or programming skills.

On the other hand, there is need for a database to train and test the system. The database required for the system needs to have both the color data and the depth measurements for every captured pixel, as well as a document defining which objects are contained together with their location in pixel coordinates.

The system specifies a top-down approach, the usage of BERKELEY 3D Object detection and positive detection rate.

1.4. Methods and procedures

The topic of this thesis is focused in the Computer vision field, in the Chair of Robotics and Embedded Systems. This Chair, has different projects related to visual tracking and person recognition, on image and video processing and also, some experience working on 3D object recognition and 3D reconstruction based on the depth data captured by the sensor on a Kinect camera.

Although work has been done in this field before, this bachelor thesis is not a continuation of a previous project and starts from the scratch with a new approach. But the knowledge and tools can be used to support the work.

1.5. Work Plan

WP Name: Research on depth estimation and multiple view geometry	WP ref: 1
Major constituent: Research	
Short description: Kinect cameras enable the reconstruction of the scene by using an IR Depth sensor. This work package consist of the research of this sensor usage, the mathematical operation behind it, how does it work and evaluate whether it is the best option for estimate depth, as well as, find the most proper way for reconstruction and understand the geometry of it.	Planned start date:10/05 Planned end date:31/05

Table1. Work Package 1

WP Name: Research on 3D-based object recognition and databases	WP ref: 2
Major constituent: Research	
Short description: The work in this package consist of the research of resources and papers about object recognition using 3D data to design the approach that is going to be used in this project and research on the most suitable dataset of images, sequences and 3D data both for reconstruction and recognition.	Planned start date:10/05 Planned end date:31/05

Table2. Work Package 2

WP Name: Development of own approach (Conceptually)	WP ref: 3
Major constituent: Development	
Short description: Definition of the approach for the project development .Resource usage, databases, libraries and any additional tool or code.	Planned start date:01/06 Planned end date:15/06

Table3. Work Package 3

WP Name: Modification/Adaptation of own approach	WP ref: 4
Major constituent: Development	
Short description: Modification on the approach previously defined in order to face the difficulties or problems that appear during the implementation of it.	Planned start date:23/06 Planned end date:05/08

Table4. Work Package 4

WP Name: Features Extraction	WP ref: 5
Major constituent: Development	
Short description: This package is one of the most extensive including the comparison and selection of the best features for both color and depth data.	Planned start date:15/06 Planned end date:05/08

Table5. Work Package 5

WP Name: Training	WP ref: 6
Major constituent: Development	
Short description: Obtaining models for different objects based on the common features of them	Planned start date:5/08 Planned end date:25/08

Table6. Work Package 6

WP Name: Testing and obtaining results	WP ref: 7
Major constituent: Development	
Short description: Different tests with different parameters and thresholds in order to obtain the optimum classifier and compare the importance of every feature.	Planned start date:25/08 Planned end date:15/09

Table7. Work Package 7

WP Name: Evaluation and comparison of own approach	WP ref: 8
Major constituent: Evaluation	
Short description: Evaluate the results obtained and verify whether the specifications are correct and if the expected results and goals are accomplished.	Planned start date:15/09 Planned end date:20/09

Table8. Work Package 8

WP Name: Presentation	WP ref: 9
Major constituent: Presentation	
Short description: Oral presentation of the work done.	Planned date:23/09

Table9. Work Package 9

WP Name: Continuous Documentation	WP ref: 10
Major constituent: Documentation	
Short description: Written documents of the work that is been doing.	Planned start date:10/05 Planned end date:20/09

Table10. Work Package 10

1.6. Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
2	1	Database comparison	Document	18/05
10	2	Project Proposal and work plan	Document	20/05
10	3	Project critical review	Document	28/06
10	4	Bachelor thesis	Document	22/09

Table11. Milestones

1.7. Gantt Diagram

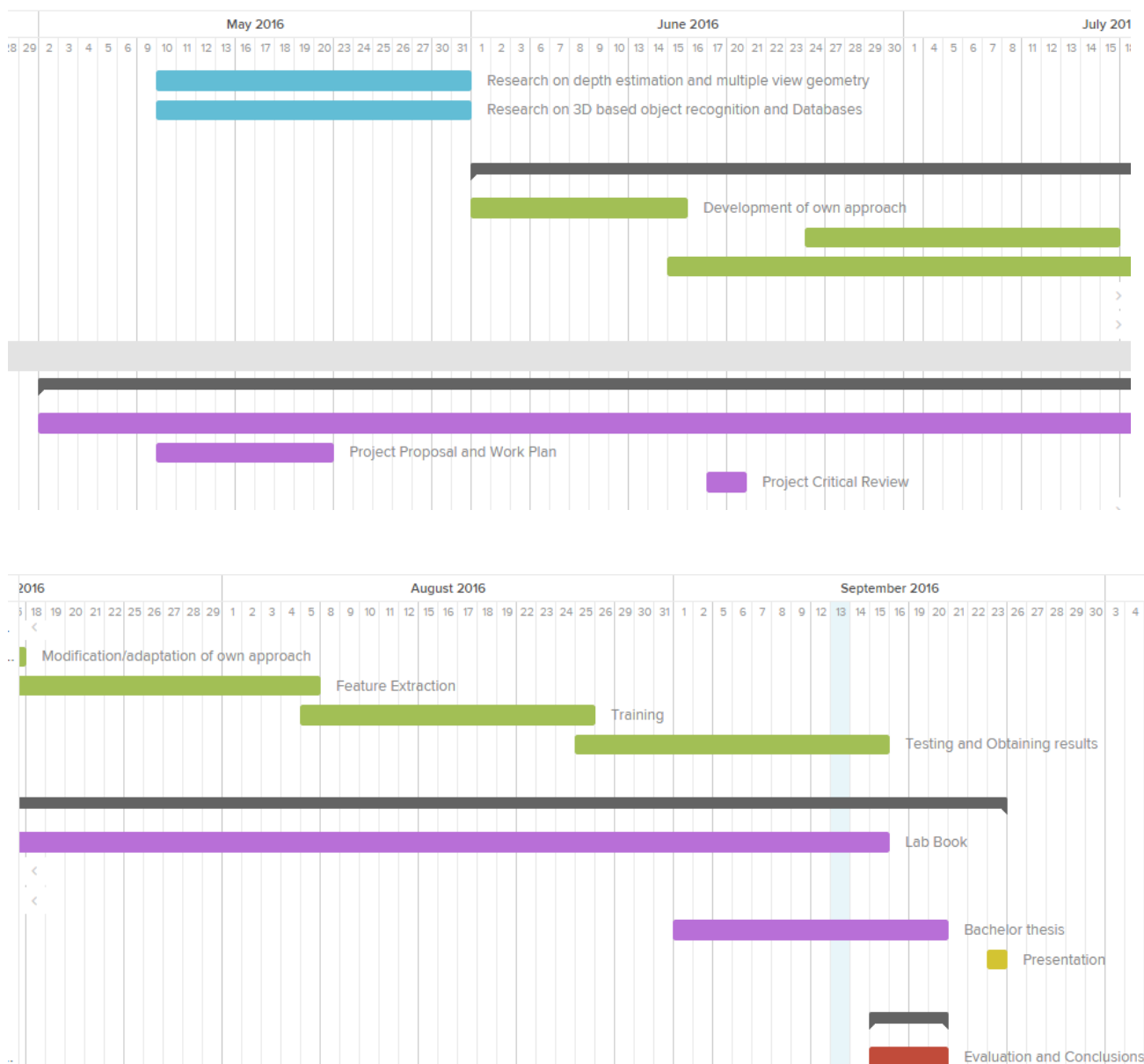


Figure1. Gantt Diagram

1.8. **Deviations and incidences**

As part of every project, it is not until we begin to develop the algorithms, when we find some difficulties or incidences to face. This, usually requires of thinking on a different approach for the problem, or adapt the actual system, which may cause a delay.

Taking this into account, during the development of the algorithms for the extraction of features, some methods were added to the initial design because it was noticed an improvement on the results. First, the plane detection on the point Cloud of the scene and secondly, the reconstruction of the objects after the plane detection, since it was seen that few pixels were overlooked.

2. State of the art of the technology used or applied in this thesis:

2.1. Kinect sensor and Depth Estimation

A Kinect sensor, is a RGB-D camera that allows to capture the depth of an image giving extra information of how far the objects are. This technology, was built to enable people to physically interact with a game with their own body using human body-language understanding. Even though this has always been an active field of research in computer vision, nowadays Kinect technology is often used in Robotics and embedded systems to create 3D figures.[1]

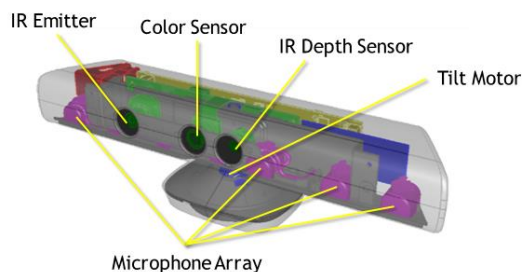


Figure 2. Scheme of a Microsoft Kinect Sensor [2]

In order to create the depth image, the **IR Emitter** of the camera, projects a pattern of infrared light to a room. As the light hits a surface, the pattern becomes distorted and this distortion is read by the **IR Depth Sensor** which will create the 3D map. The **Color Sensor** provides the color information.[2]

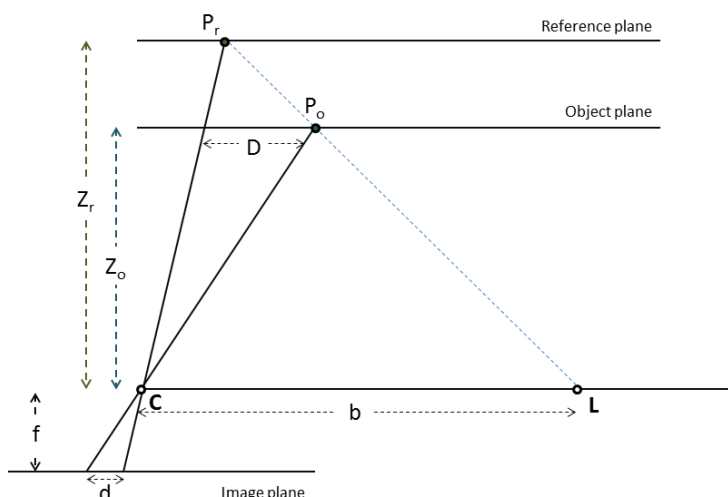


Figure 3. Trigonometry for depth calculation [3]

In figure 3, $\mathbf{P_r}$ is the position of a point in a reference depth $\mathbf{Z_r}$. $\mathbf{P_o}$ is the same point captured by the Kinect at a depth $\mathbf{Z_o}$, the depth we want to calculate. \mathbf{D} is the 3D disparity between the 2 points, while \mathbf{d} is the disparity on the 2D image plane. \mathbf{f} is the focal length, and \mathbf{b} is the distance between the camera \mathbf{C} and the IR Emitter \mathbf{L} . [3]

Using trigonometry, the depth of the object can be calculated as:

$$Z_o = \frac{Z_r}{1 + \frac{Z_r}{fb}d}$$

[3]

2.2. Databases

The state of the art in computer vision has rapidly advanced over the past decade and the cheap but quality depth Kinect sensor has brought the possibility of building datasets for a challenging category-level 3D object detection. Thus, the main idea behind the construction of these datasets, is to allow the validation of 3D surface reconstruction methodologies. [4][5]

The good quality of existing image datasets do not include 3D or pose information, while other 3D datasets lack in variation of scenes, categories, instances, and viewpoints.

The database used in this project is the Berkeley 3-D Object Dataset, taken in domestic and office settings. The sensor provides a color and depth image pair.

The first release of the dataset contains 849 images taken in 75 different scenes. Over 50 different object classes are represented in the crowd-sourced labels. The annotation is done by Amazon Mechanical Turk workers in the form of bounding boxes on the color image, which are automatically transferred to the depth image. [6]

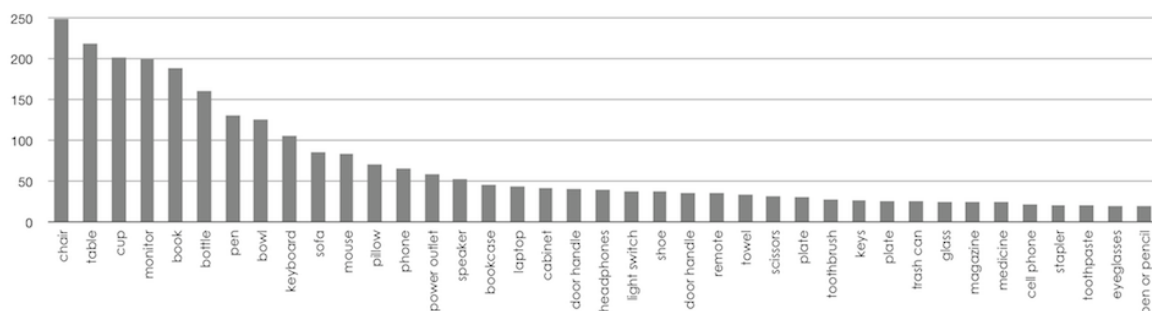


Figure 4. Number of representations for every object in the Database. [6]

2.3. Object Recognition and Design of own approach

Recognizing objects is not a new area of research. Object recognition methods have always employed a training phase. For example, to build a detector of objects belonging to the same class, we first need a lot of pictures of it in order to create a model: flexible in shape, size, color or every other feature that defines it, so we can find similarities in future objects we want to detect. So basically, pattern recognition and machine learning are needed.

This Bachelor thesis, is going to be based on the work done by researchers of different universities, and described in the paper published by H.Ali in 2013.[7]

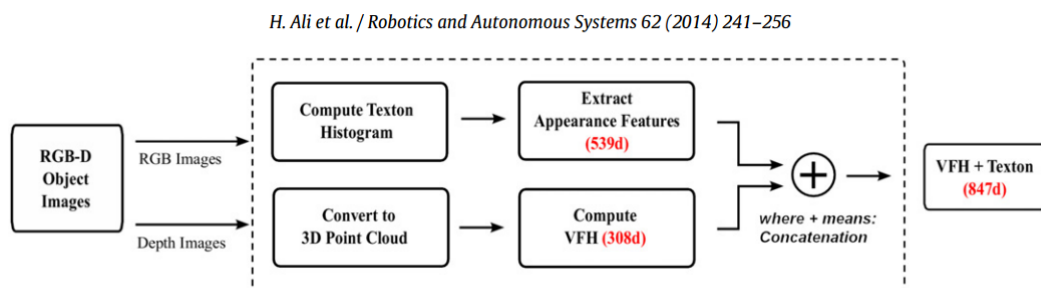


Figure 5. Scheme for object recognition [7]

The algorithm for object recognition relies on the scheme in *Figure 5*. The database used, provides separately the colour and the depth image. So, the main idea is to extract the corresponding features from each one. For the color data, a color histogram is needed while for the depth information a previous step is also needed: the generation of the point cloud.[7].

2.4. Point Cloud Data

A Point Cloud is a set of points represented in a three-dimensional coordinate system.

A recent initiative in this area is PCL (Point Cloud Library) created to provide support for all 3D building and 3D perception. It is also meant to incorporate 3D processing algorithms like filtering, feature estimation, surface reconstruction, model fitting, segmentation, registration, etc. [8]

In this project, the first functionality required will be the creation of the Point Cloud. For this, it is important to know the geometry of the camera used.

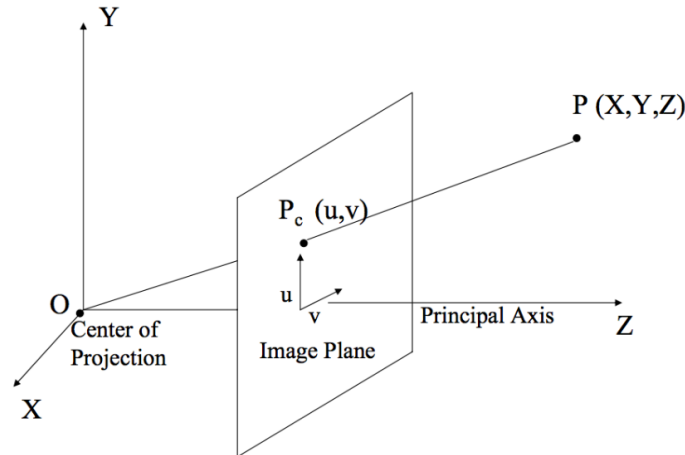


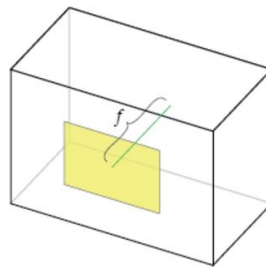
Figure 6. Ideal pinhole camera model [9]

The ideal pinhole camera model describes the relationship between a 3D point $(X, Y, Z)_T$ and its corresponding 2D projection (u, v) onto the image plane.

The projection of a 3D world point $(X, Y, Z)_T$ onto the image plane at pixel position $(u, v)_T$ can be written as

$$u = \frac{X \cdot f}{Z} \quad \text{and} \quad v = \frac{Y \cdot f}{Z}$$

f denotes the focal length. The focal length is the distance between the pinhole and the image plane, measured in pixels.



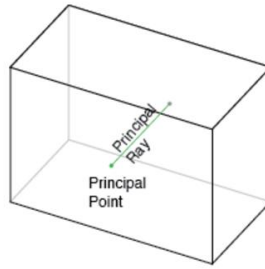
[9]

To avoid such a non-linear division operation, the previous relation can be reformulated as:

$$\tilde{\lambda} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Most of the current imaging systems define the origin of the pixel coordinate system at the top-left pixel of the image. However, it was previously assumed that the origin of the pixel coordinate system corresponds to the principal point $(O_x, O_y)_T$, located at the center of the image.

The camera's "principal axis" is the line perpendicular to the image plane that passes through the pinhole. Its intersection with the image plane is referred to as the "principal point".[9][10]



A conversion of coordinate systems is thus necessary. Using homogeneous coordinates, the principal-point position can be readily integrated into the projection matrix. The perspective projection equation becomes now:[9][10]

$$\tilde{\lambda} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & x_o & 0 \\ 0 & f_y & y_o & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{cases} \tilde{\lambda} \cdot u = f_x \cdot X + x_o \cdot Z \\ \tilde{\lambda} \cdot v = f_y \cdot Y + y_o \cdot Z \\ \tilde{\lambda} = Z \end{cases}$$

$$\begin{cases} X = \frac{\tilde{\lambda}(v - y_o)}{f_y} \\ Y = \frac{\tilde{\lambda}(u - x_o)}{f_x} \\ Z = \tilde{\lambda} \end{cases}$$

[9]

Secondly, most of the robots developed lately are meant to move in an indoor space on a horizontal surface. So, it is really important to have a good knowledge of the planes contained in the image such as walls, floors, doors, etc. to detect easily the objects laying to these surfaces.[11]

2.5. Fast Point Feature Histogram

The Viewpoint Feature Histogram(VFH) is a descriptor for 3D point cloud data that encodes both geometry and viewpoint. It has been demonstrated experimentally that can be used as a distinctive signature, coping well noise levels and different sampling densities, allowing simultaneous the recognition of the object and its pose.

In this project, we are going to use the simpler version of this descriptor, the Point Feature Histogram (PFH). The goal of the PFH formulation is to encode a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values. Subsequently, the histogram collects the pairwise pan, tilt and yaw angles between every pair of normals on a surface patch.[12]

The general algorithm to compute this histogram includes:

For each point p in a cloud P

- 1- *Get the nearest neighbors of P .*
- 2- *For each pair of neighbors, compute the three angular values.*
- 3- *Bin all the results in an output histogram.*

Usually, a number of 16 neighbors allows to get good results. To compute the difference between two points $\langle p_t, p_s \rangle$, and their associated normals $\langle n_t, n_s \rangle$, we define a fixed coordinate frame at one of the points:

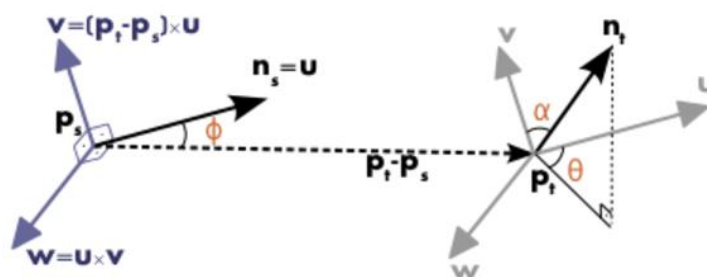


Figure 7. Model of PFH for a pair of points[12]

$$\begin{cases} u = n_s \\ v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|^2} \\ w = u \times v \end{cases}$$

Using the above uvw frame, the difference between the two normals n_s and n_t can be expressed as a set of angular features as follows:

$$\begin{cases} \alpha = n_t \cdot v \\ \phi = u \cdot \frac{(p_t - p_s)}{\|p_t - p_s\|^2} \\ \theta = \arctan(w \cdot n_t, u \cdot n_t) \end{cases}$$

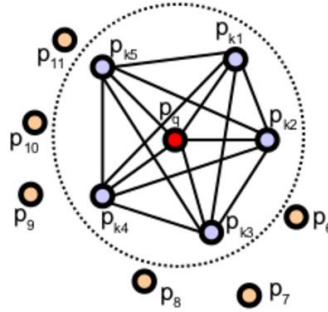


Figure 8. Model of a region diagram PFH [12]

This Figure below presents a region diagram of the PFH for a query point (p_q), placed in the middle of a circle, and all its k neighbors. The final PFH descriptor is computed as a histogram of relationships between all pairs of points in the neighborhood.

Computationally PFH is complex, specifically $O(n^2)$ in the number of surface normals n . In order to make a more efficient algorithm, we will take advantage of the usefulness of the Fast Point Feature Histogram. The FPFH measures the same angular features as PFH, but estimates the sets of values only between every point and its k nearest neighbors, reducing the computational complexity to $O(k * n)$. [12]

SPFH (Simplified PFH) between the key point and every neighbor:

$$FPFH(p_i) = SPFH(p_i) + \frac{1}{k} \sum_{j=1}^k \frac{1}{w_j} \cdot SPFH(p_j)$$

where the weight w_j , represents a distance between the point p_i and p_j . [13]

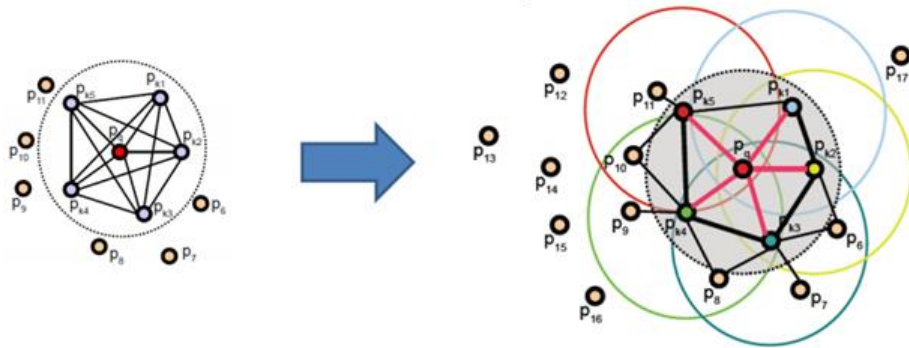


Figure 9. Model of a region diagram FPFH [13]

To create the final SPFH representation for an specific point, the set of all angular features is binned into a histogram.

This process divides each angular feature's value range into b subdivisions, and counts the number of occurrences in each subinterval. One option is to divide each feature interval into the same number of equal parts, and consequently create a histogram with b^3 bins. In this space, a histogram bin increment corresponds to a point having certain values for all its 3 features. [13]

3. Methodology / project development:

3.1.FEATURES EXTRACTION

The database used in this project contains, for every image, the RGB information and the corresponding depth data, obtained from the Kinect camera. In order to conduct the recognition of the objects that are going to be trained, some features need to be extracted. In this case, information both from colour and depth will be used, following the algorithm mentioned in section 2.3. The first step, consists of an interpretation of the data in the XML file of the image we want to deal with. The XML file would provide the information of how many objects are in the scene and the coordinates of the bounding boxes around them.

After this first step, the algorithm is divided into two different branches. On the one hand, a segmentation is computed on the RGB image (as explained in section 3.1.1). This is useful to separate the pixels corresponding to the object from the pixels of the background. After transforming these pixels of colour into grayscale values, we obtain the histogram that would define the color properties of the object.

On the other hand, using the depth map, we create a point cloud of the scene (section 3.1.2). Secondly, using the position of the points in the 3D environment, the planes of the scene are eliminated in order to keep just the objects. The planes usually belong to walls, floors or background. Comparing the resulting point cloud and the one corresponding to the bounding box of the object, we finally obtain the true positive points: in other words, the points of the object we want to define. In some cases, we lose parts of the object in the plane detection (section 3.1.3), for example when the object is on a table. That is why, a reconstruction step (section 3.1.4) needs to be performed in order to retrieve this data. Finally, we create a vector feature histogram (section 3.1.5), the feature that would allow to differentiate one object from the rest.

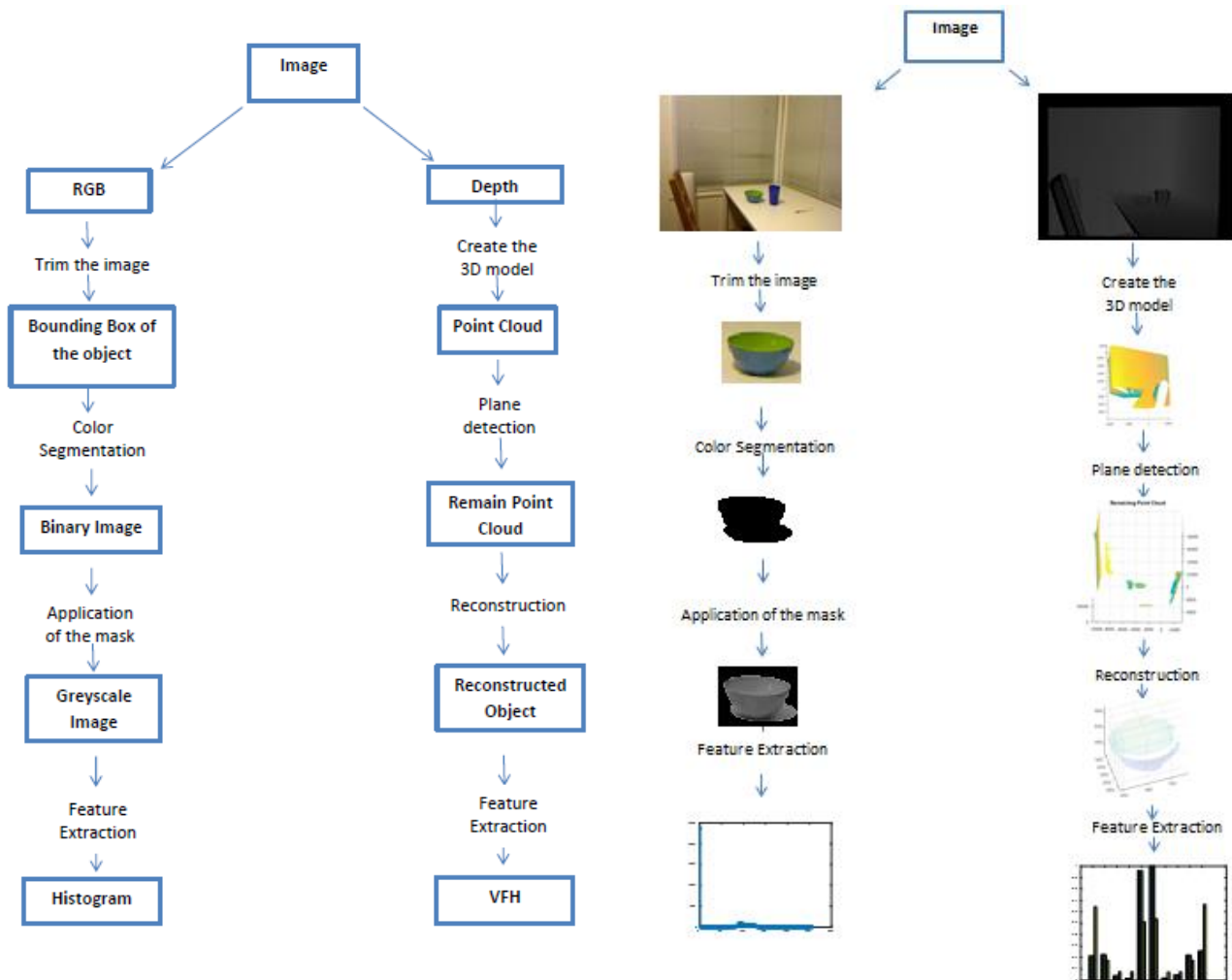


Figure 10. General scheme for feature extraction

3.1.1 Colour Segmentation

In order to separate the pixels corresponding to the object from the pixels belonging to background, we need to segment the image. Different methods already exist for image processing, such as clustering, watershed segmentation, or texture filters. Nevertheless, after the evaluation of the images response from Berkeley Database to different methods, we can conclude the best results were obtained using an opening method (verified in section 4.1).

This method uses a morphologic structuring element to detect the objects: the opening filter. This filter combines morphologic dilation and erosion in order to eliminate the small objects that do not belong to the object we want to detect, and reduce the noise of the image. The element we use has to be larger than the element we want to eliminate.

The algorithm followed to obtain this binary mask of the original image performs the following steps: first, the background estimation in order to subtract the background image from the scene containing the object. After that, we increase the image contrast creating a new binary image by thresholding the adjusted image.[14]

$$Y = (X \ominus b) \oplus b$$

$$\begin{cases} X = \text{Original Image} \\ b = \text{Structural Element} \end{cases}$$

EROSION: $A \ominus B = \bigcap_{b \in B} A_{-b}.$

DILATATION: $A \oplus B = \bigcup_{b \in B} A_b,$

Once we have segmented the image, we need to binarize it. We are going to use this binary mask to select the pixels from the original image that we consider part of the object.

In order to finally create the histogram defining the colour information, we need to transform the RGB pixels in a grey scale.

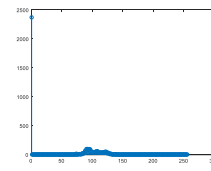


Figure 11. Example of Colour Segmentation: From left to right: a) RGB image, b) Binary mask obtained after segmentation c) grey pixels of the object d) histogram

3.1.2 Point Cloud Creation

The information required for this step consists of: the depth map and the intrinsic parameters of the Kinect camera used for capturing the images. The information provided from these parameters, and the formulas mentioned in the previous section 2.4, allow the transformation of the 2D image into its 3D representation. Every parameter defining a geometrical property of the camera is provided in Table 12 below. Figure 12, shows an example of the 3D representation (point cloud) obtained.

$$f_x = 368.096588$$

$$f_y = 368.096588$$

$$x_0 = 261.696594$$

$$y_0 = 202.522202$$

Table 12. Intrinsic Parameters of Depth Camera (Kinect v1) [15]

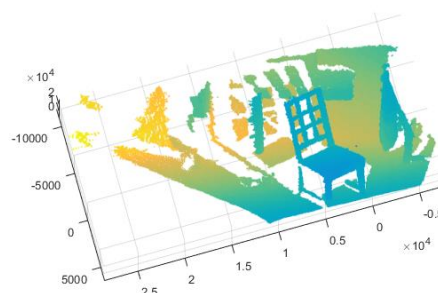
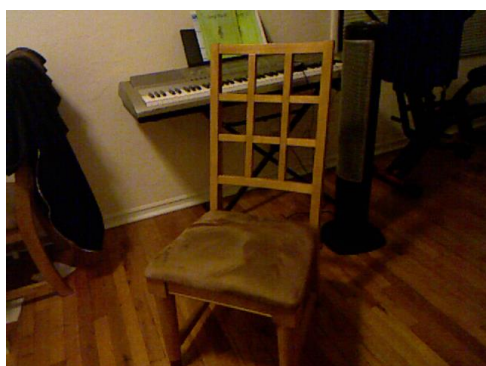


Figure 12. Example of Point Cloud: a) RGB Image b) Corresponding Point Cloud

3.1.3 Plane Detection

Due to the increasing size and complexity of geometric data sets, the recognition of objects in these scenes has become a difficult task. Moreover, Kinect cameras have a lot of noise that make even more complicated to recognize the geometry of the objects.

One approach for point clouds that can help improving the results, is the detection of shapes. In our case, we have a large variety of objects, all of them of different size and shape. However, in our database all images are indoor. This is very beneficial for this step because we can consider homogeneous backgrounds: walls, floors etc. Moreover, in some of the cases we have the desired object on a table, a uniform surface.

Taking all of this into account, we implement an algorithm based on the work mentioned in the section 2.4, capable to detect every plane of the cloud, and eliminate it. [11].

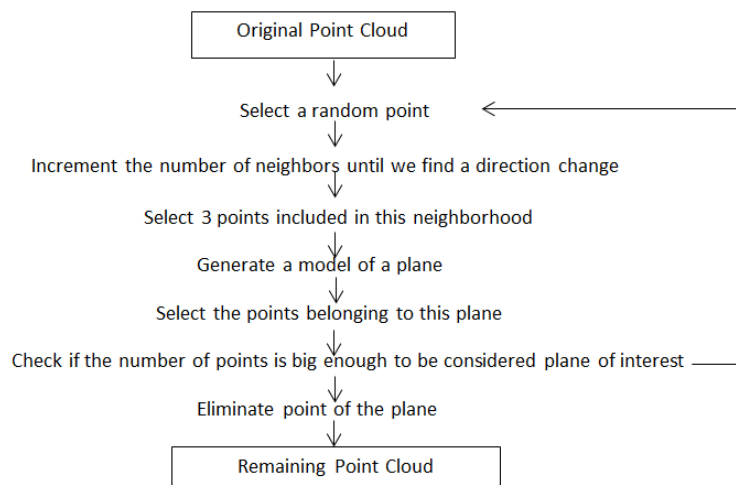


Figure 13. Scheme of plane detection

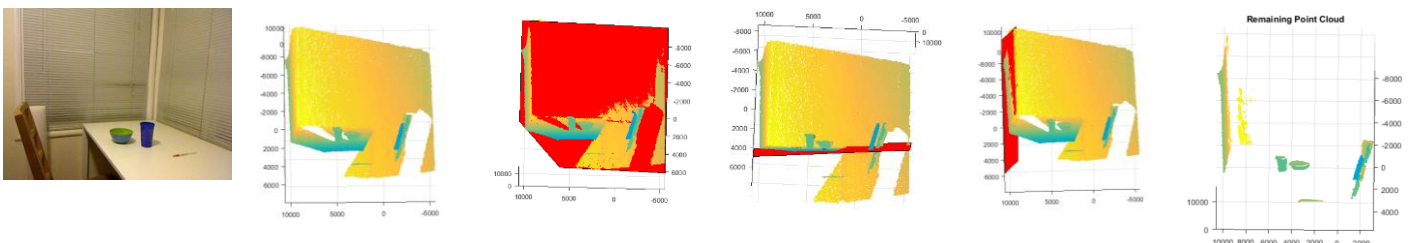


Figure 14. Example of Plane Detection: a) RGB image b) Point cloud c)d)e) plane detected f) Remaining point cloud

3.1.4 Object Reconstruction

Once the planes had been removed from the point cloud of the image, the resulting cloud contains the detected objects and some noise produced in previous steps.

Clustering this result, would allow recognizing how many groups of pixels belong to an object or, in the other hand, whether they are erroneous points and consequently have to be deleted or simply ignored. For this task, the information contained in the XML of every image will be very useful.

The data we find in the XML specifies the pixels of the bounding boxes for every image. Therefore, this approach will start by creating the point cloud of the box containing the object which we are working. Comparing this result with the remaining point cloud we have stored, we obtain finally the 3D representation of the object.

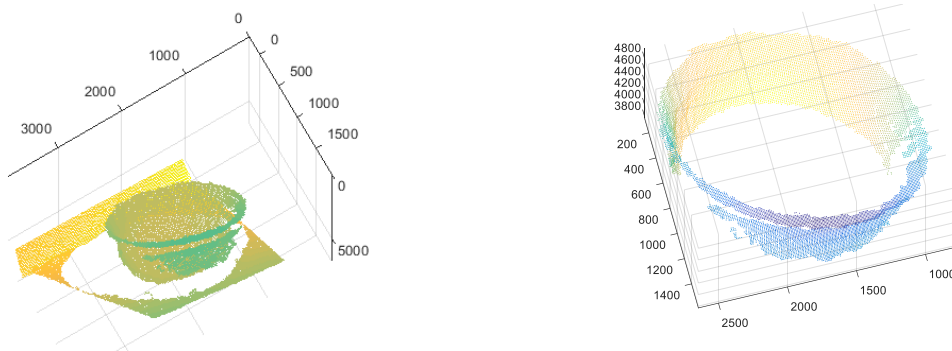


Figure 15. Example of Remaining Point Cloud
a) Point cloud of the bounding box containing a bowl
b) Common points from a) and the remaining point Cloud of the scene

At this point, an optional step could be applied to improve the result. In some cases, when the size of the objects is not big enough, or if they are too close to a wall, an important loss of valid pixels can occur during the plane extraction.

At this point, we compute the reconstruction of our object. This function includes an examination of the planes extracted, in the interest of finding that many lost points, considering the Euclidean distance of them to the center of the object.

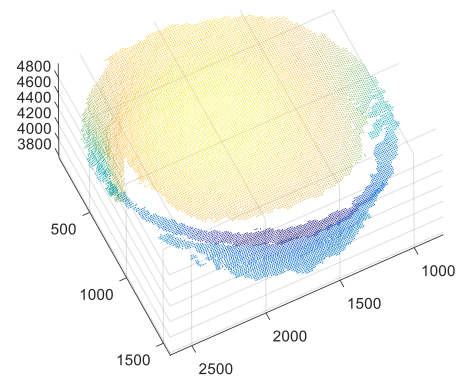
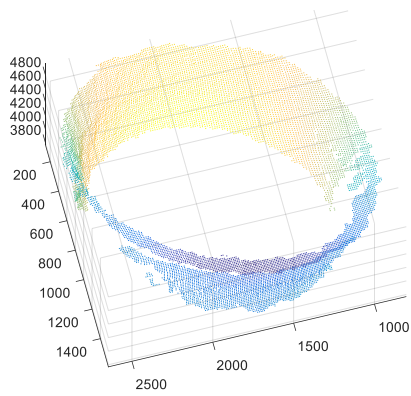


Figure 16 Example of Object Reconstruction .a)Point cloud of a bowl
b) Reconstructed version

3.2. TRAINING AND TESTING

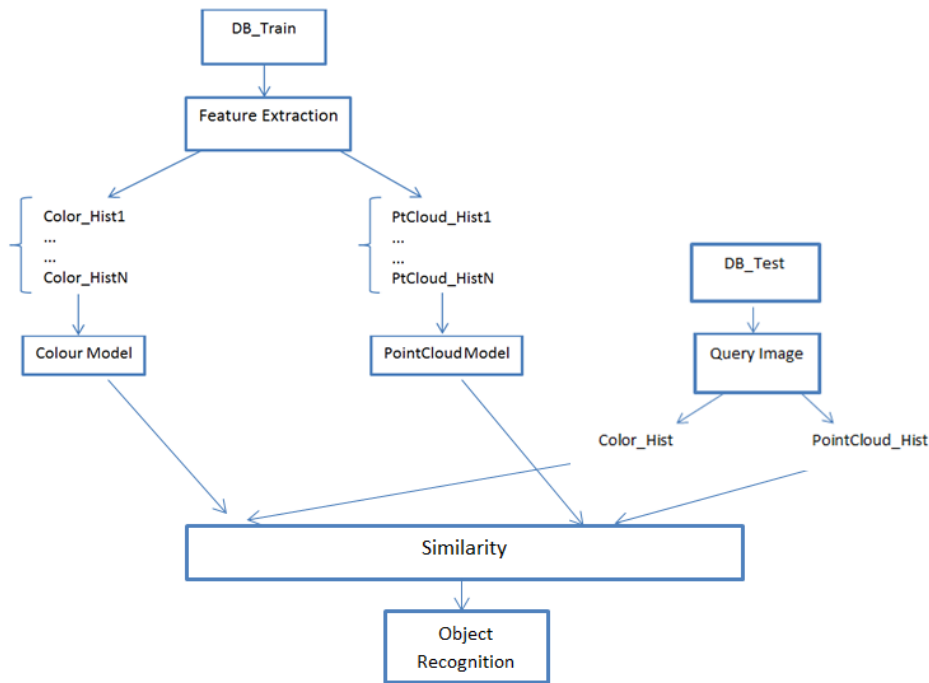


Figure 17. General scheme for testing

To accomplish the recognition of the objects, the first step is to divide the general database into two different parts: the training and the testing. The training part will be used to create a model for identification, in other words, compute for every object both a histogram of the colour features and the histogram representing the features of the Point Cloud (FPFH). The model is created to have a general description of the object we want to detect.

The second block of images is part of the test database, these objects will permit the evaluation of our recognition system. For every object the same color and point cloud histograms will be computed. Comparing this result with the model generated for every kind of object, we can designate it to a class or another depending on the similarity of them.

In this project, 70% of the original database is used for training while the remaining 30% is assigned to testing.

3.2.1 Test Database

This created database contains 30 objects, all from different images.

- 10 bowls
- 10 pillows
- 10 monitors
- 2 cans
- 1 chair
- 1 book
- 1 purse
- 1 bookcase
- 1 phone
- 1 sofa
- 1 bike helmet
- 1 bottle

3.2.2 Test algorithms

The database used for this system, Berkeley DB [6], contains 849 images taken in 75 different scenes and over 50 different object classes are represented. Due to the large volume of data, the testing part will be divided in different algorithms, starting with the most simple classification, more objects and conditions can be added to improve the results of it.

The first algorithm just compares the object we are testing, and a model, deciding if it is the same object or not, while the second one takes into account more than one model. A third method, not implemented, would not use an object as a query but directly an image, so a segmentation needs to be done to detect the objects of the images and then the testing can be computed.

3.2.2.1 Testing a model

In this first testing, we will test just one model. To do that, we measure the distance of the given histogram and the histogram of the model. Whether the distance exceeds the threshold, we consider it valid or no.

3.2.2.2 Testing more than one model

In this second method, more models are added to be tested. In this case, having an object as a query to identify, we compute the distance of it to all the models we have trained. The object will be assigned to the most similar class, however, to improve the results we also take into account the thresholds defined in the first algorithm, so we can also detect the objects that don't belong to any of the trained classes.

3.2.3 Evaluation of the classifier

In order to evaluate the results of the system and check the quality of the classifiers, we will use a receiver operation characteristic or ROC curve. The curve is created by plotting the true-positive rate, known as recall, against the false positive rate or probability of false alarm in machine learning.

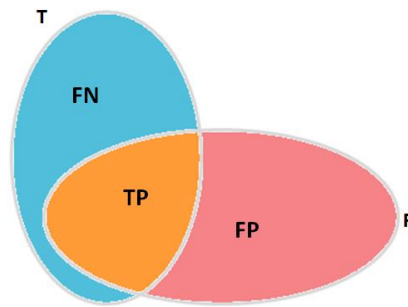


Figure 18. Scheme for the evaluation of a classifier

In our case, we use it to check if the objects have been correctly classified, for example, one of the first and simplest test corresponds to detect if an object is a bowl or not. As an example, the first circle(T) in *Figure 18* would represent all the bowls while the second one would contain the objects that the system had classified as bowls. So, in the intersection of both circles we would get the true positive, while the blue region would have the ones we skipped. Finally the pink one would represent the objects the system classified as bowls but they are not.[16]

TP: Classify the bowls as bowls

FP: Classify as bowls other objects

FN: Not to recognize that the object is a bowl

The ROC space is defined by the Recall (TP/T) as x axis: fraction of correct samples (T) that have been correctly detected, and the probability of false alarm (FP/F) as y axis: fraction of false samples (F) that have been detected wrong. The best result is when the the line gets the upper left corner (0,1) representing that all the objects have been correctly classified.

By defining a diagonal that divides the ROC space, we can assure the points above the diagonal represent good classification, meanwhile as more points below it we have, the poorest is the prediction.[16]

The accuracy of the system is defined as:

$$ACC = \frac{\sum TP + \sum TN}{\sum Total Population}$$

[16]

4. Results

4.1. RESULTS OBTAINED IN THE FEATURES EXTRACTION

4.1.1 Colour Segmentation

In order to separate the pixels corresponding to the object from the pixels belonging to background, we need to segment the image. For this step, two different methods were tested using the Berkeley Database, in order to compare results and to estimate which method obtains more satisfactory results.

4.1.1.1 Opening Method

This filter allows to select the shape and the size of the element we are going to apply for the segmentation.

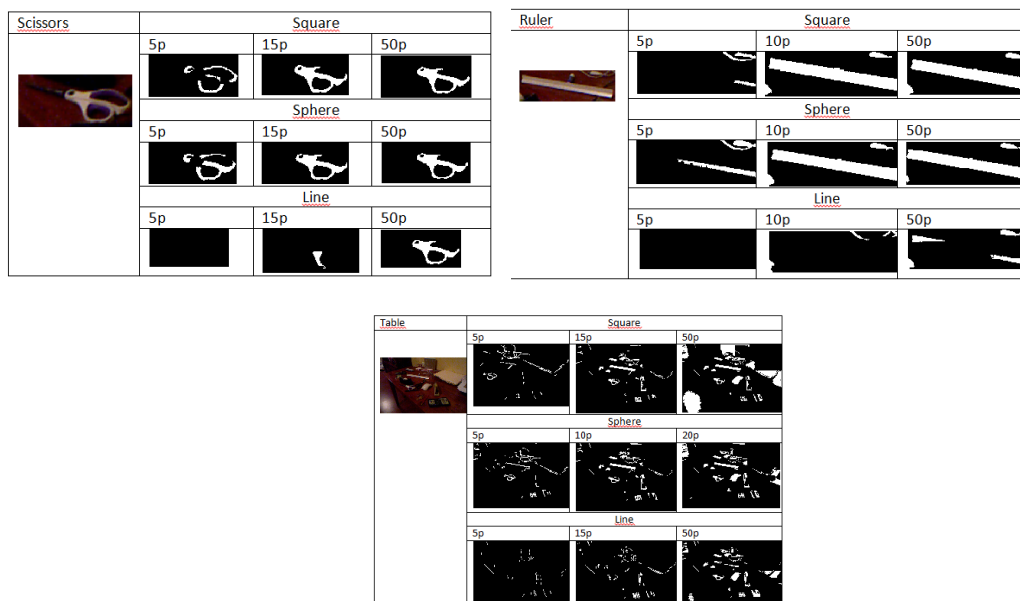


Figure 19. Examples of segmentations using different elements and shapes

In these examples, we can observe that the elements that obtain better results are the 'square' and the 'sphere'. The 'Line' filter could be useful if the tilt of the object is known beforehand, otherwise the result is not good, such as in the case of the ruler image.

This method is useful to eliminate the small objects that do not belong to the object we want to detect and to reduce the noise of the image. In some cases, like the ruler, the results are quite good using only a square or a sphere of 10 pixels as the element, or even with the scissors we obtain almost the whole object and we can recognize it. However, using this method with the table, the results in the binary mask are rather poor.

4.1.1.2 K-means Clustering

The algorithm consists of classifying a given data set through a certain number of clusters fixed a priori. The main idea is to define k centers, one for each cluster. So, the choice is to place them as far as possible from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. At this point we need to re-calculate k new centroids. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center.

In order to test this method, the same images were used. However, in this case, the changing parameters were the number of clusters and the number of loops.

		nClusters		
Scissors	nLoops	N=2	N=3	N=5
	N=3			
	N=10			
	N=30			

		nClusters		
Ruler	nLoops	N=2	N=3	N=5
	N=3			
	N=10			
	N=30			

		nClusters		
Table	nLoops	N=2	N=3	N=5
	N=3			
	N=10			
	N=30			

Figure 20. Examples of segmentations using different numbers of Clusters and Loops

This second method allows to improve some results, such as the case of the table. However, in other simple cases like the ruler, the result is not satisfactory enough. Moreover, the computational complexity of the k-means Clustering is much bigger, slowing the performance of the algorithm. That is why in this procedure, we decided to use the Opening Method.

4.1.1.3 Results of the final segmentation

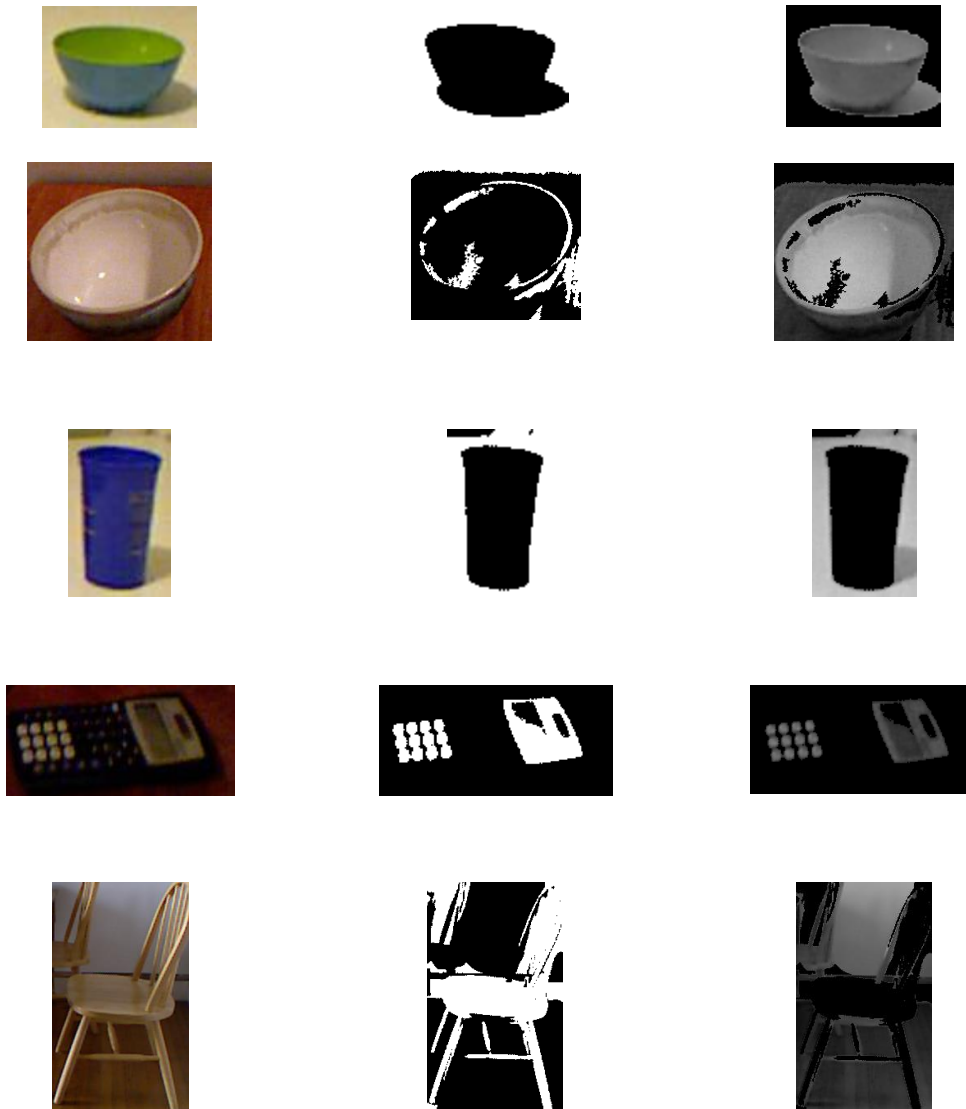


Figure 21. From left to right: a) RGB image,
b) Binary mask obtained after segmentation
c) grey pixels of the object

As we can observe in the examples shown in *Figure 21* this chosen segmentation has very good results in some cases (the bowl or the cup), but in some other objects such as the calculator, we experiment a loss of some pixels due to the variability of colour and the similarity to the background.

4.1.2 Point Cloud Creation

In this step, the information of depth is represented as a point Cloud for every image. The results are successful. Nevertheless, the easy detection of the objects will depend of the position of the camera and the overlap in the distribution of the room, imperfections arising from the depth data when capturing the images.

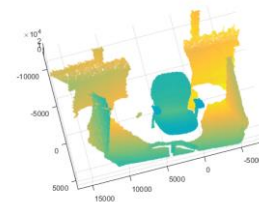
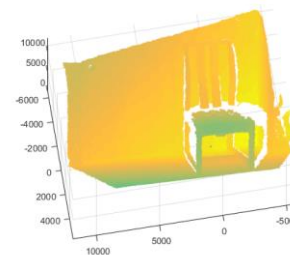
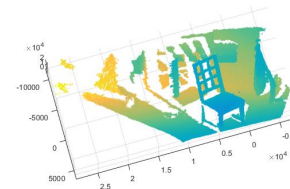
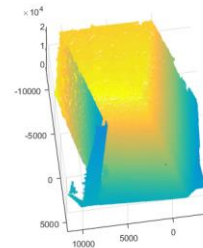
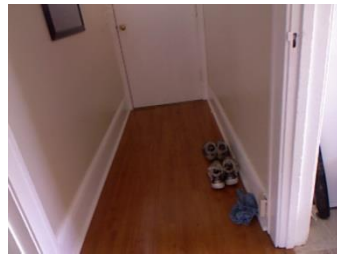
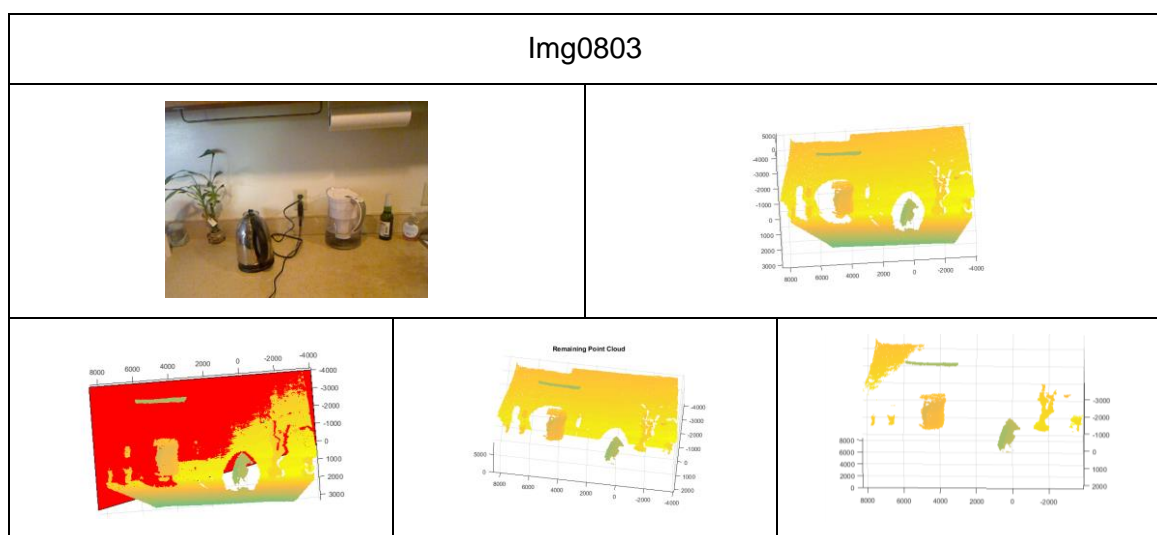
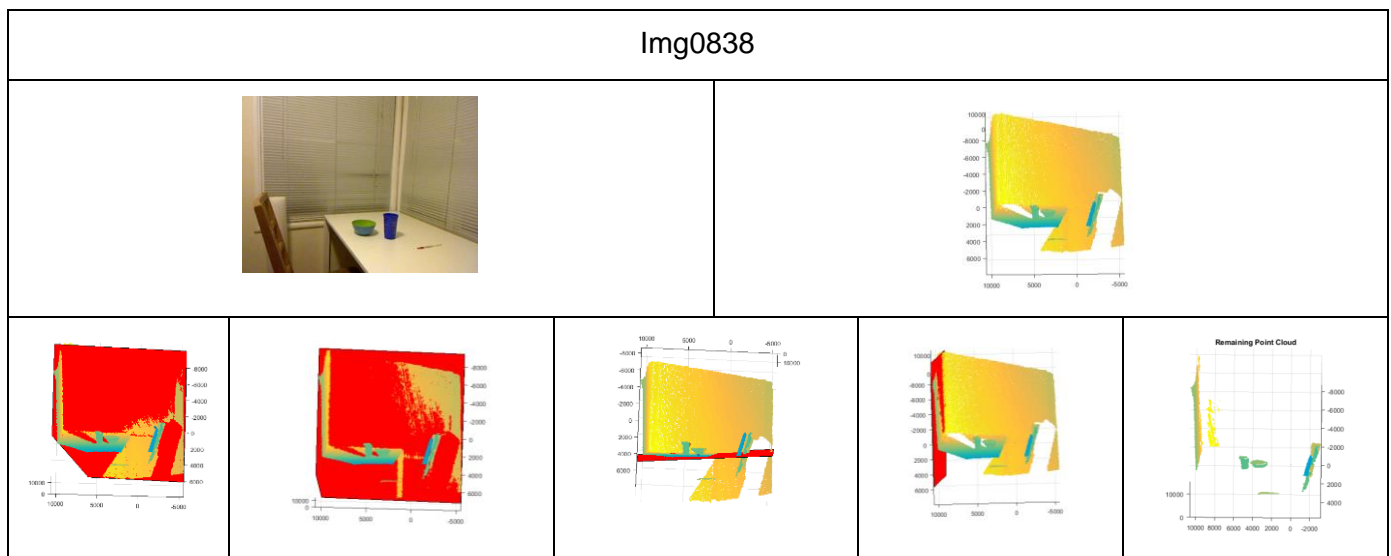


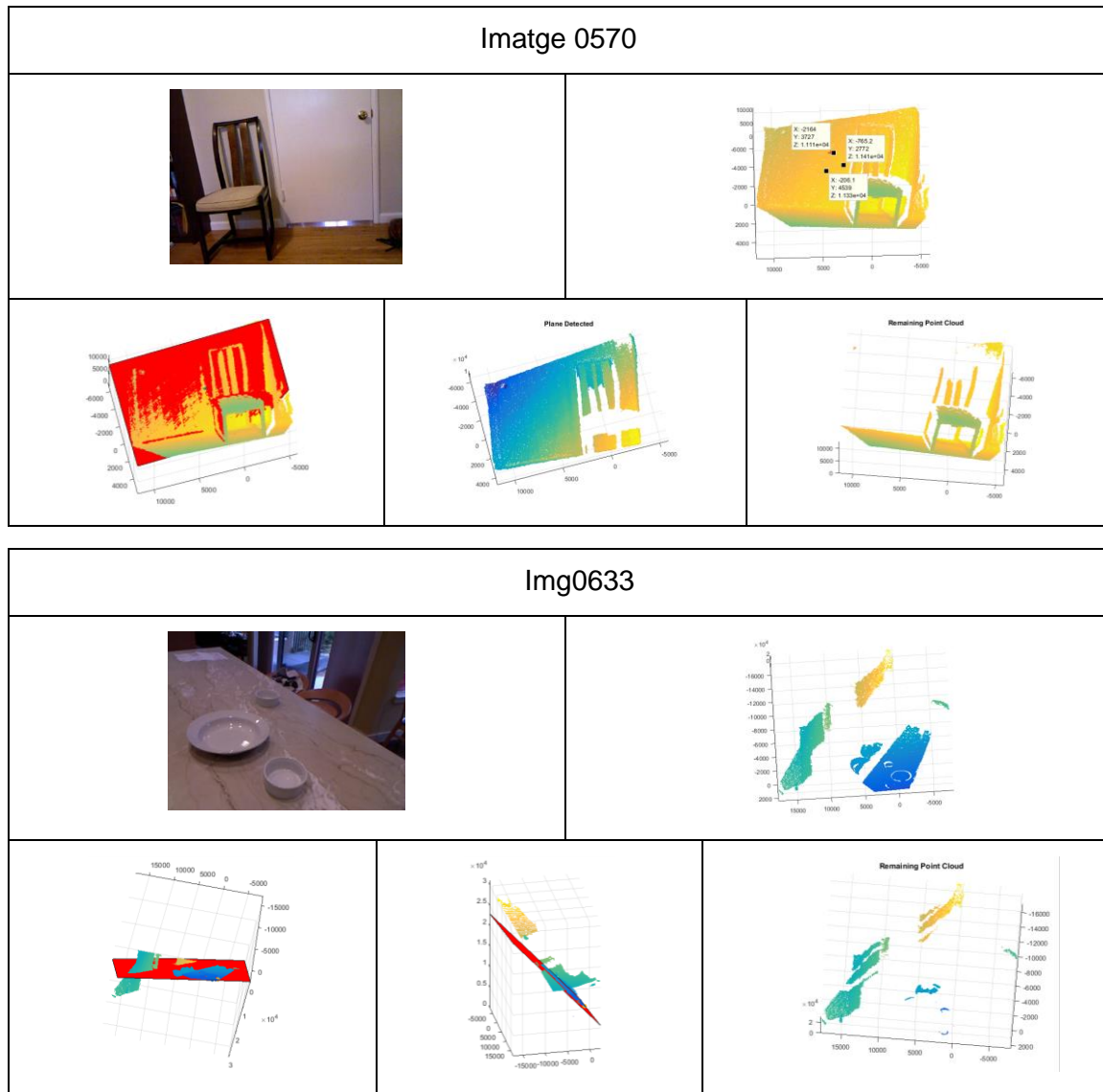
Figure 22. Results for the Point Cloud Creation. From left to right: a) RGB image, b) Point Cloud representation

4.1.3 Plane Detection

The algorithm for the plane detection starts with the selection of a random point. That means that every time we execute it the result can change, and different planes can be detected. Moreover, the time for this algorithm is limited, so that if most of the random points, selected during this fixed duration, do not correspond to any plane, this may lead to the skipping of some planes.

Another problem related to this step is that for the objects lying on a plane or close to it, it may lose part of the pixels when the plane is eliminated. However, this loss will be fixed in a following step, explained in section 4.1.4.





*Figure 23. Results for the Plane Detection: From up left to bottom right. A)RGB image
b)Point Cloud c)Plane Detection
d)Remain Point Cloud*

The plane detection in the images 0803 and 0838 is very satisfactory, since in the final Point Cloud, once the planes have been eliminated, we obtain perfectly the objects we wanted to detect without any loss.

On the other hand, in the image 0570 we experiment one of the mentioned inaccuracies, due to the fact that the system has been able to eliminate the wall, but the plane of the floor is still present in the final Point Cloud.

Finally, the last example, image 0633, shows how part of the object, in this case a plate and bowl, have lost some of the pixels when the plane has been eliminated.

4.1.4 Object Reconstruction

In order to fix this last inaccuracy when the planes are eliminated, we compute an object reconstruction. The problems we can face in this procedure are: on the one hand, the lack of true positive pixels and, on the other hand, the reconstruction of parts that don't belong to the object.

The function created for this step includes an examination of the planes extracted, in the interest of finding the lost points, considering the Euclidean distance of themselves to the center of the object. A threshold will define which distance can be considered object and which one can be discarded.

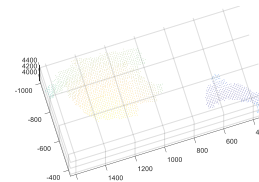
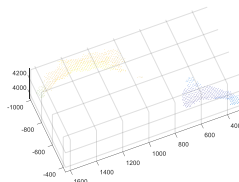
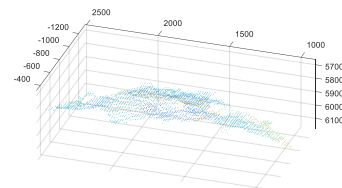
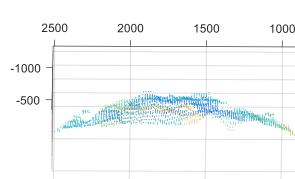
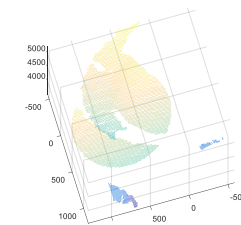
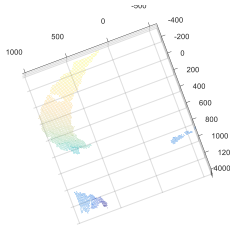
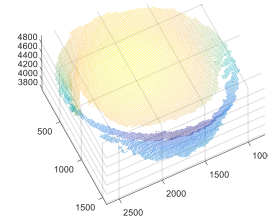
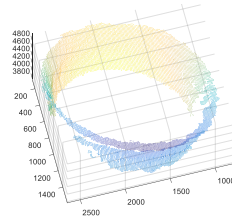


Figure 24. Results for the Object Reconstruction: From left right. A) RGB image b) Detected object c) Reconstructed Object

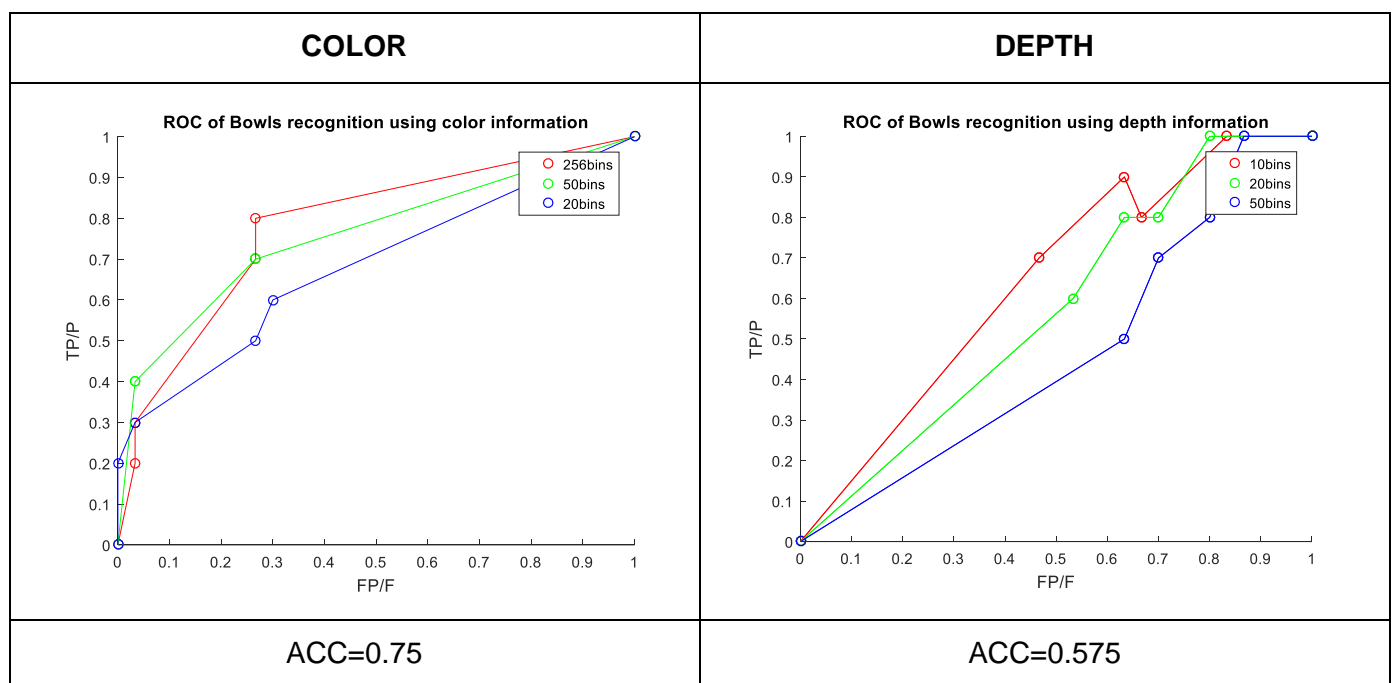
This reconstruction step can be very useful in some cases, like the first bowl, since we can reconstruct a large part of the object. However, in other cases, for example the bike helmet, the detected object and its reconstruction are quite similar.

4.2 TESTING RESULTS

4.2.1 Testing a model

In this section we test every model separately, classifying every object from our test database as positive, when the similarity of it and the model is below the threshold or as negative if the distance between both histograms exceeds this threshold.

This experiment is executed several times with different parameters: the number of bins and the thresholds selected.



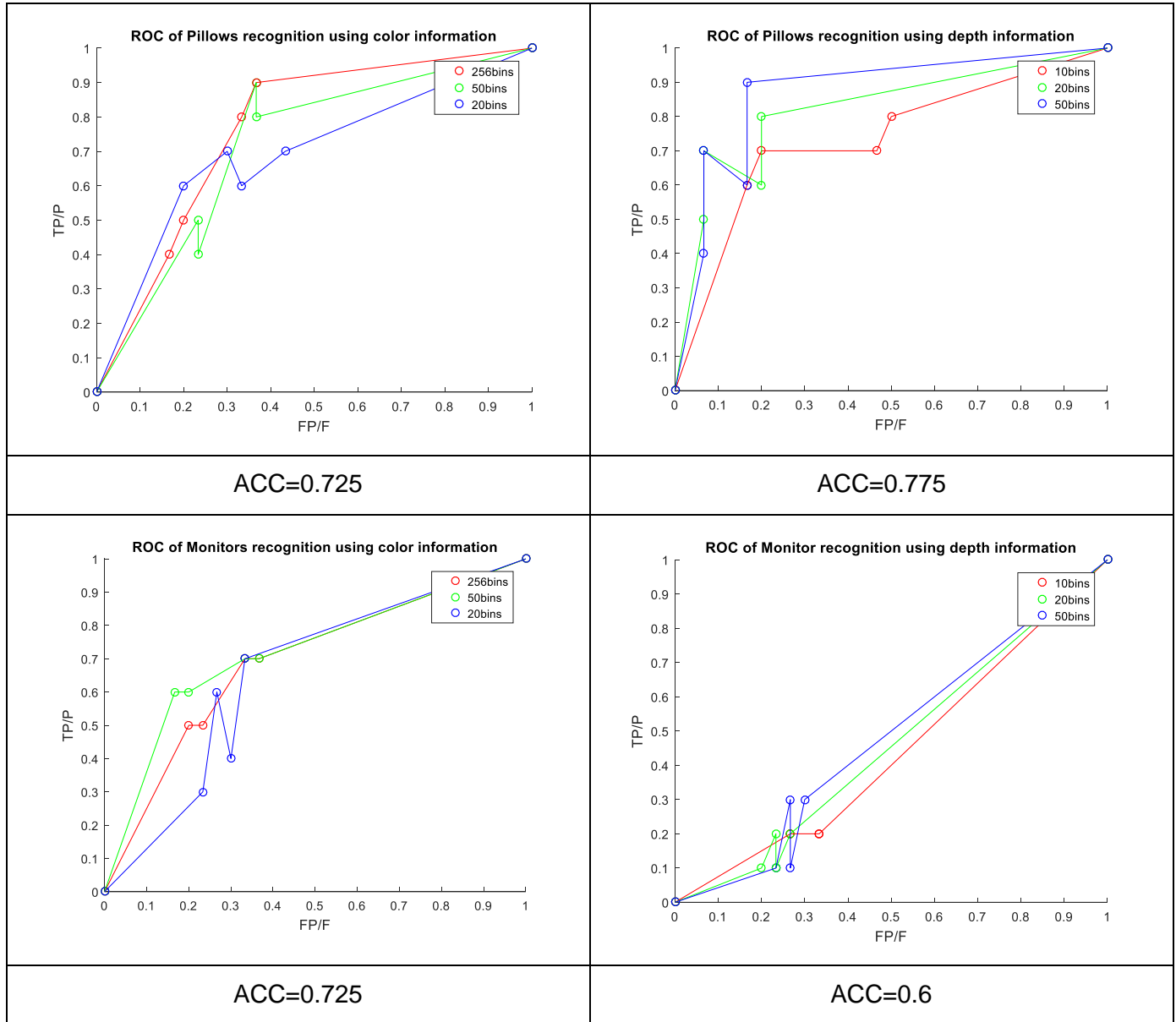


Figure 25. Results test1. From left to right: a) Using different bins for color
b) Using different bins for depth

The figures on the left side, represent the ROC curves for every model using only the color features, while in the figures on the right, we can observe the same curves but in this case, depth features are used.

For the first case, using color, the same classification has been done three times comparing the results using different bins: 256, 50 and 20. As the images are represented normally with 256 gray-level, using all the bins, the result is the optimum. As we start to quantize the result, combining levels into a lower number of bins, the theory says the result gets worse.

Studying the behavior of the 3 objects, using these 3 levels of bins, we can analyze if the behavior is the expected one. For every experiment, we use 4 different thresholds from more restrictive to more permissive. In the example of the Bowls, we can conclude that the number of bins does not really affect in one of the attempts, however, in all the others, the more bins we use the better the result is.

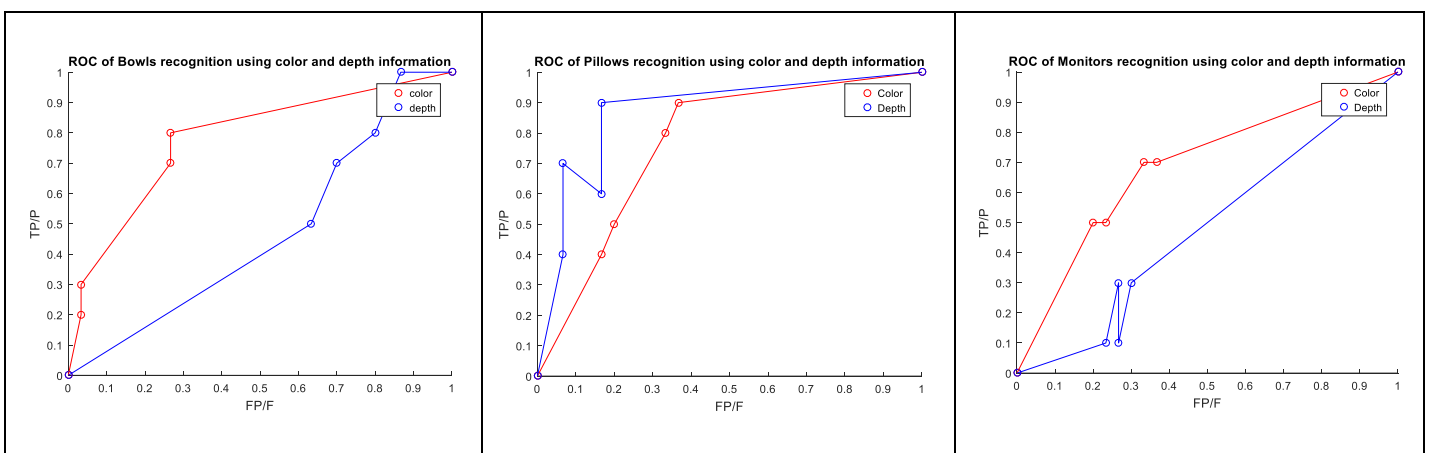
In the case of the Pillows, the response of the ROC is similar, obtaining the best results with 256 bins. Nevertheless, in the last case, testing the pillows, 50 bins have resulted in higher accuracy.

Similar steps have been done using the depth. In this case, the number of bins used is: 10, 20 and 50. The expectations are that results improve as more bins we use. This is true for Pillows and Monitors. However, for the Bowls detection we get a better result using just 10 bins.

In general, the outcomes using the color features are slightly superior. However, it can vary in some objects. The first case, the bowls, the poor result using the depth data, may be because of the size of the object, sometimes covered for other objects in the scene, that lead to the incomplete reconstruction of it.

The second example, the pillow, the accuracy using the depth data is higher (0.775). Moreover, the classification of this object is the best out of the 3 objects.

Finally, the monitors get a pretty good result in the color classification as in the most of the representations the monitor is black. However, the results are poorer in the depth, perhaps due to the similarity in the shape of pillows in some images, to the monitor model.



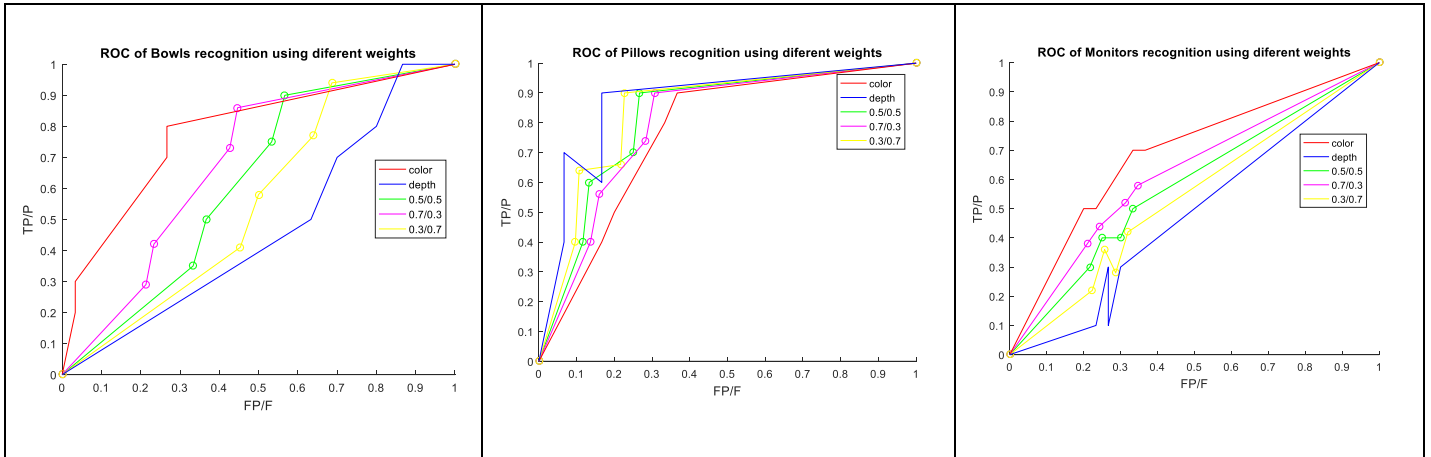


Figure 26. Combination of Color and Depth Features. From left to right:
a) Bowls b) Pillows c) Monitors

Using the best results from the previous step (256 bins for Color and 50 for Depth) the next task consists of combining these two outcomes, to get a complete classifier.

In the first three graphs of the Figure 26, the red line represents the ROC using the color while the blue ones refer to depth information. In the second line of the same Figure 26, the diagram shows 3 different representations using different weights assigned to every classification. In green, the division is the same, 50% each feature. The pink line gives more importance to the color, specifically 70%, and finally the yellow one gives priority to the depth data, also 70%, remaining just 30% for the color.

We can observe that, in every case the combination of both classifiers always worsen the outcome. Nevertheless, the result is still acceptable and the accuracy remains positive in all cases.

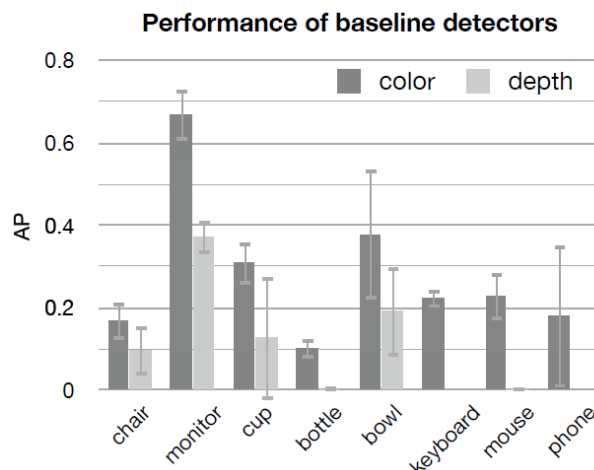


Figure 27. Performance of the baseline detector described in another project[6].

It is not that easy to compare the results obtained here, with previous algorithms developed in other projects, because the number of objects used is not the same as well as usage of different variables in the detection procedure. However, we can compare, for example the work by “Max-Planck-Institute for Informatics” [6], *Figure 27*. For this work, the database used, was the BERKELEY DB as well, and the same objects were studied. The y-axis, represents the average precision: the true positive samples, divide the positive ones. We can observe that in this case, the color representation is always better than the depth. Also, the results for the monitor are higher than the bowl. The bowl, gets values above 0.3 for the color and above 0.2 for depth. Comparing this numbers with *Figure 27*, we can see the results obtained are higher.

4.2.2 Testing more than one model

Improving the previous testing, in this second method, more models are added to be tested. The object is assigned to the most similar class, as long as it remains to the margins imposed previously.

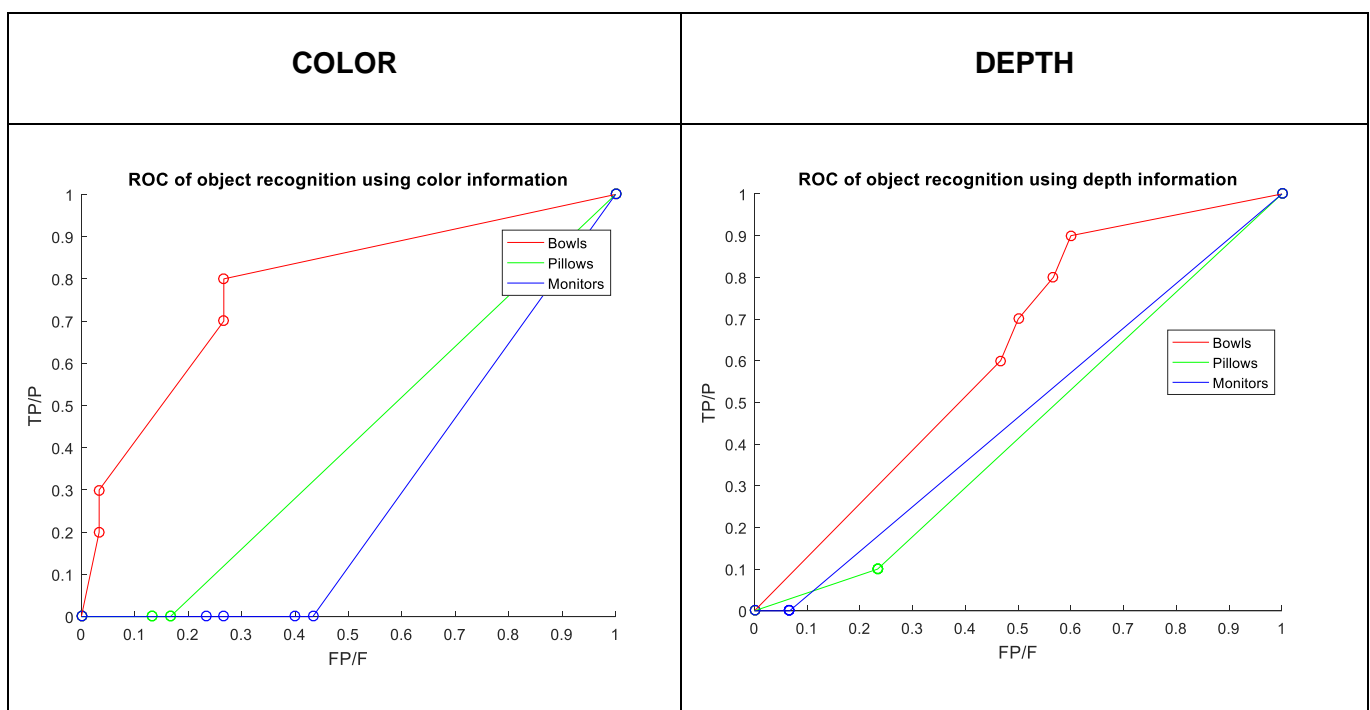


Figure 28. Results for test2.

Testing all the results at the same time leads to a decrease of the quality of the classifier with respect to the results of the first test. The first object, the bowl, is still good detected both using color and depth with accuracy above the 50%. Even though, in the classification of other objects, such as the monitors and pillows, this parameter takes much lower values. As more objects are added, the worst the result. This is because the

more options we have to compare, more likely it is that the system can confuse it with another object. For example, at this point the shape and the size of bowls and pillows are really different. However, if we introduce bottles or dishes to the system, the color features will be really similar to the features of the bowl, increasing the number of errors.

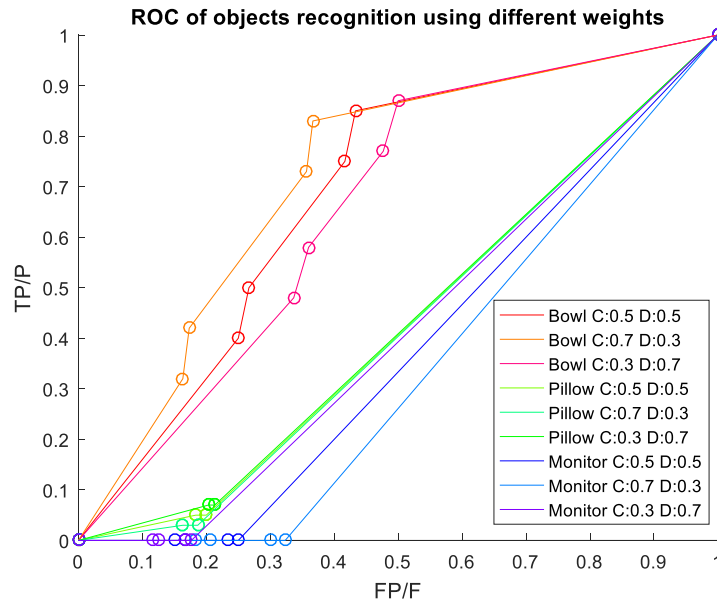


Figure 29. Results for test2 using different weights

This Figure 29, shows again 3 different representations using different weights assigned to every classification. Even though in some cases the difference is almost negligible, like in the case of the monitor, in other cases, for example the bowl, the classification is much better when the balance leans towards the color features.

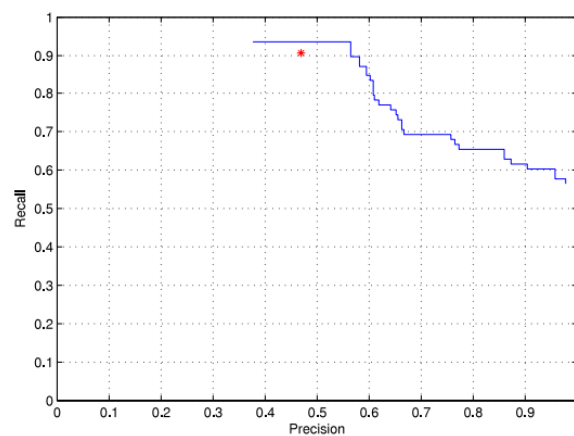
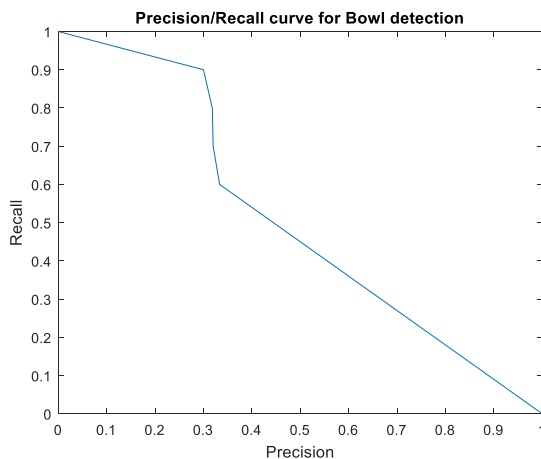


Figure 30. Comparing Precision/Recall Curves a) Own project b) Project [7]

In this second test, we compare the results obtained in our project with the ones obtained in previous work [7]. For this purpose, we compute the precision recall curve for the bowl detection. The quality of our classifier is lower than the other approach. However, in our case we compare 3 different models, while the second curve represents the response of the bowl detection but for a binary classifier.

5. Budget

This project is not a prototype, but an exploration study on the effectiveness of object detectors in 3D. It was entirely developed in MATLAB, for which the standard license price is 2000€

The number of hours dedicated to the development has been a total of 25 hours per week, for a period of 21 weeks: 525 hours. The cost of a junior engineer is around 9€ per hour, being the final cost 4725€.

In total, the price of the project is 6725€.

MATLAB standard license	2000€
Cost of a junior engineer	4725€
TOTAL	6725€

Table 13. Budget of the project

6. Conclusions and future development:

We can finally conclude that the recognition of 3D objects is not easy. Even though the technology of depth cameras has improved remarkably lately, the generation of precise models using depth information from commercial depth sensors is still not precise enough. It can be used in a large variety of applications, however, for medical procedures or other strict robotic tasks, the results are not yet adequate.

Taking a look into the goals of this project, it can be said that the objectives have been achieved, having as a result a classifier of objects using both color and depth information. Regarding the results obtained, it can be said that the colour information, and therefore the features related to it, provide much better results, creating a more robust classifier. However, we should not consider a classifier based only on color data an accurate or trustable detector. For example, for the pillow detection, we can have a large variety of colors, as this object can have multiple and really different colors. And also, an object completely different, such as a chair, could be the same color.

This is one of the main problems found during the development of this project, so that is why the future implementations should focus on the improvement of 3D objects and depth information: since the reconstruction of partial objects for the training, to the discard of erroneous points considered part of the object. Also, this project has been developed using a Kinect of first generation. So, another way to improve this data is to use a second version Kinect that would provide a depth channel with less noise.

For the testing final step, this bachelor thesis worked directly on the recognition of objects, previously extracted from the image. The following task in this section, is to work directly with an image, first segmenting the point cloud to detect how many objects appear in the image, to later apply the object recognition algorithms on them.

Bibliography:

- [1] W.Zeng. "Microsoft Kinect Sensor and Its Effect". *University of Missouri*, 2012. [Online] Available: <https://www.microsoft.com/en-us/research/publication/microsoft-kinect-sensor-and-its-effect/> .
- [2] A.Davudinasad. "Kinect Sensor". *Amirkabir University of Technology (Tehran Polytechnic)* [Online], ResearchGate Available: https://www.researchgate.net/publication/268517966_Kinect_Sensor
- [3] K.Khoshelham. "Accuracy analysis of Kinect depth data" *University of Twente*. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-5/W12, 2011 ISPRS Calgary 2011 Workshop, 29-31 August 2011, Calgary, Canada
- [4] A.Doumanoglou, S.Asteriadis, Dimitros S.Alexiadis. "A Dataset of Kinect-Based 3D scans" Information Technologies Institute, Centre for Research and Technology Hellas, 6th km Charilaou Thermi, GR-57001, Thessaloniki, Greece
- [5] A.Singh, J.Sha, K.S.Narayan, T.Achim, P.Abbeel "BigBIRD A Large-Scale 3D Database of Object Instances" Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94709
- [6] A.Janoch, S.Karayev, Y.Jia, J.T.Barron, M.Fritz, K.Saenko, T.Darrell "A Category-Level 3-D Object Dataset: Putting the Kinect to Work" UC Berkeley and Max-Plank-Institute for Informatics
- [7] H.Ali, F.Shafait, E.Giannakidou, A.Vakali, N.Figueroa, T.Varvadoukas, N.Mavridis. "Contextual object category recognition for RGB-D scene labeling". *Robotics and Autonomous Systems* 62. 2014.
- [8] R.Bogdan, Steve Cousins. "3D is here: Point Cloud Library (PCL). Willow Garage USA.
- [9] "Dissecting the Camera Matrix, Part 3: The Intrinsic Matrix" 2013 [Online] Available: <https://ksimek.github.io/2013/08/13/intrinsic/>
- [10] Y.Morvan. "Acquisition, Compression and Rendering Depth and Texture For Multi-View Video" Ph.D.Thesis 2009 Multi-view video coding Publications.
- [11] R.Cupec, R.Grbic, K.E.Nayarko, K.Sabo, R.Scitovski. "Detection of Planar Surfaces Based on RANSAC and LAD Plane Fitting" University of Osijek 2009. Croatia.
- [12] PCL Documentation[Online] Available: http://pointclouds.org/documentation/tutorials/pfh_estimation.php
- [13] F.Tombari "Keypoints and Features" Pisa 2013[Online] Available: http://www.pointclouds.org/assets/uploads/cglibs13_features.pdf
- [14] Sheetal, R.P.Goela, K.Garg."Image Processing in Hand Vein Pattern Recognition System" International Journal of Advanced Research in Computer Science and Software Engineering V4, Issue 6, June 2014 ISSN:2277 128X.
- [15] Depth-color registration parameters [Online] Available: <https://github.com/OpenKinect/libfreenect2/issues/41>
- [16] T.Fawcett. "An introduction to ROC analysis" *Pattern Recognition Letters* 27. 2006 Institute for the study of Learning and Expertise, USA.

Glossary

2D	2 Dimensions
3D	3 Dimensions
PCL	Point Cloud Library
ROC	Receiver Operating Characteristic
PFH	Point Feature Histogram
VFH	Viewpoint Feature Histogram
FPFH	Fast Point Feature Histogram
WP	Work Package
IR	Infrared
XML	eXtensible Markup Language
ACC	Accuracy
TP	True Positive
FP	False Positive
FN	False Negative