Universidade de Aveiro
2015

Departamento de Eletrónica,
Telecomunicações e Informática

Rui Pedro Figueiredo Garcia

Reconhecimento de objetos para um robô de serviços

Object recognition for a service robot

Universidade de Aveiro
2015

Departamento de Eletrónica,
Telecomunicações e Informática

Rui Pedro Figueiredo Garcia

Reconhecimento de objetos para um robô de serviços

Object recognition for a service robot

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António José Ribeiro Neves, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Paulo Miguel de Jesus Dias, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais.

o júri / the jury

presidente / president

Professor Doutor Armando José Formoso de Pinho
Professor Associado C/ Agregação, Universidade de Aveiro

vogais / examiners committee

Doutor Miguel Armando Riem de Oliveira
Bolseiro Pós-Doutoramento, Inesc - Porto (Arguente Principal)

Professor Doutor António José Ribeiro Neves
Professor Auxiliar, Universidade de Aveiro (Orientador)

agradecimentos /
acknowledgements

Aos meus orientadores, Professor Doutor António Neves e Professor Doutor Paulo Dias, o meu mais sincero obrigado pela disponibilidade e constante acompanhamento que me ofereceram ao longo deste último ano.
À minha namorada Inês, pelo carinho e motivação com que me acompanhou durante esta fase, bem como a paciência e compreensão que sempre demonstrou para comigo.
Ao Fred, pelo companheirismo e boa disposição com que sempre me acompanhou durante o desenvolvimento deste projeto.
Também aos meus companheiros de longa data, em especial ao Eduardo, Tiago, Bruno, Fábio e Daniel, o meu obrigado por toda a amizade e confiança que depositaram em mim ao longo desta fase da minha vida.
Aos meus pais e irmão, pois sem eles nada teria sido possível.

Palavras Chave

Visão por computador, deteção e reconhecimento de objetos, segmentação 3D.

Resumo

A contínua evolução da tecnologia e o crescimento no desenvolvimento de aplicações robóticas tornou possível a criação de robôs autónomos que consigam assistir ou até mesmo substituir os humanos em tarefas diárias e trabalhos monótomos. Atualmente, com o envelhecimento da população humana, é esperado que os robôs de serviço venham a ser cada vez mais utilizados para assistência de pessoas idosas ou com deficiência. Para isso, um robô de serviços tem que ser capaz de evitar obstáculos enquanto se movimenta em ambientes conhecidos ou desconhecidos, ser capaz de detetar e manipular objetos e perceber comandos dados pelos humanos. O objetivo desta dissertação é o desenvolvimento de um sistema de visão, capaz de detetar e identificar objetos, para o robô CAMBADA@Home. O sistema de visão proposto implementa dois métodos para deteção de objetos, sendo o primeiro baseado em histogramas de cor e o segundo método usando algoritmos de deteção e descrição de pontos de interesse (algoritmos SIFT e SURF). O sistema usa informação de profundidade e de cor, sendo a informação 3D usada para detectar objetos que estejam pousados sobre superfícies planas. Os resultados experimentais obtidos com o robô CAMBADA@Home são apresentados e discutidos, com o objetivo de avaliar a robustez do sistema proposto.

Keywords Computer vision, object detection and recoginition, 3D segmentation.

Abstract

The continuous evolution of technology and the fast development of robotic applications has made possible to create autonomous robots that can assist or even replace humans in daily routines and monotonous jobs. Nowadays, with the aging of the world population, it is expected that service robots can be explored to assist elderly or disable people. For this, a service robot has to be capable of avoiding obstacles while navigating in known and unknown environments, recognizing and manipulating objects and understanding commands from humans. The objective of this dissertation is the development of a vision system, capable to detect and recognize household objects, for the service robot CAMBADA@Home. The proposed approach implements two methods for object detection, the first one based on color histograms and the second method using feature detection algorithms (SIFT and SURF algorithms). It uses depth and color information, where the 3D data is used to detect the objects that are found on horizontal planes. Experimental results obtained with the CAMBADA@Home robot are presented and discussed, in order to evaluate the robustness of the proposed system.

Contents

Contents . . . i

List of Figures . . . iii

List of Tables . . . v

1 Introduction . . . 1
1.1 Objectives . . . 2
1.2 Thesis structure . . . 3

2 Object Detection in Service Robots . . . 5
2.1 Service Robots . . . 5
2.2 RoboCup@Home . . . 7
2.3 Cambada@Home . . . 11

3 Depth Segmentation . . . 15
3.1 External Software . . . 15
3.2 Proposed approach . . . 17
3.3 Cloud pre-processing . . . 20
3.4 Plane segmentation . . . 22
3.5 Object extraction from the plane . . . 24
3.6 Results . . . 25

4 Image Segmentation using depth information . . . 29
4.1 Proposed approach . . . 29
4.2 Registration process . . . 30
4.3 Results . . . 34

5 Object Recognition . . . 37
5.1 Visual Descriptors . . . 37
5.1.1 Color Histograms . . . 38
5.1.2 SIFT - Scale Invariant Feature Transform descriptor . . . 39
5.1.3 SURF - Speeded Up Robust Feature descriptor . . . 40
5.2 Matching . . . 41
5.2.1 Color histogram comparison . . . 41
5.2.2 Descriptors matching . . . 43
5.3 Proposed approach . . . 43

5.4 Results 45
5.4.1 Study of the methods for color histogram comparison 45
5.4.2 Study of the descriptors matching algorithms 46
5.4.3 Study of the decision process 49
5.4.4 Efficiency 49

6 Experimental Results 51
6.1 Test scenario 51
6.2 Results 52

7 Conclusions and future work 57
7.1 Conclusions 57
7.2 Future Work 58

Bibliography 59

List of Figures

1.1 Point cloud captured from a robot. 2

2.1 Examples of images from different types of cameras. 6
2.2 The KeJia robot. 8
2.3 The Care-O-bot robot. 9
2.4 The AMIGO robot. 10
2.5 Dynamaid and Cosero robots. 11
2.6 CAMBADA@Home: current platform. 12
2.7 CAMBADA@Home: final platform design. 13

3.1 Point Cloud Library example. 16
3.2 OpenCV Library example. 16
3.3 Robot Operating System: communication diagram example. 17
3.4 Kinect position and orientation relative to a plane. 18
3.5 Depth segmentation: structure of the proposed approach. 19
3.6 Screenshot of a point cloud provided by the Kinect ROS driver. 19
3.7 3D voxel grid. 20
3.8 Example of a pre-processed point cloud. 22
3.9 Fitted line with RANSAC. 23
3.10 An example of a convex hull of a set of points. 23
3.11 Results of a plane segmentation. 24
3.12 Screenshot of the set of point clouds resulting from the clustering process. 25
3.13 Clustering process: retrieved clusters for each scenario. 27

4.1 Color Segmentation: structure of the proposed approach. 30
4.2 Projection result of the original point cloud to a 2D image. 32
4.3 Resulting images of the objects after the application of the ROI. 33
4.4 Example of images resulting from the projection of the clusters. 35

5.1 Two images with similar color histograms. 38
5.2 RGB histograms of the example image. 39
5.3 Example of keypoints detection using SIFT algorithm. 40
5.4 Example of keypoints detection using SURF algorithm. 41
5.5 Object Recognition: structure for the proposed approach. 44
5.6 Images database for the study of the methods for color histogram comparison. . 45
5.7 Movement described by the robot. 46
5.8 SIFT descriptors matching. 47

5.9 Number of matches for the SIFT implementation. 47
5.10 SURF descriptors matching. 48
5.11 Number of matches for the SURF implementation. 48

6.1 House environment from RoboCup@Home 2015 competition. 51
6.2 Images database for the experimental tests. 52
6.3 Example of the scenario used in the experiments: scenario 1. 53

List of Tables

3.1 Voxel Grid Filter: efficiency results. 20
3.2 Processing time of the first frame. 26
3.3 Average processing time of the remaining frames. 26
3.4 Detection ratio of the developed system. 27

4.1 Experimental results: processing time of the projection process. 34

5.1 Color histogram comparison - study of methods. 45
5.2 Experimental results: efficiency of the histogram comparison process. 50
5.3 Experimental results: efficiency of the SIFT algorithm implementation. 50
5.4 Experimental results: efficiency of the SURF algorithm implementation. 50

6.1 Experimental results: efficiency of the developed system using the SIFT algorithm. 53
6.2 Experimental results: Recognition rate regarding the SIFT algorithm. 54
6.3 Experimental results: efficiency of the developed system using the SURF algorithm. 54
6.4 Experimental results: Recognition rate regarding the SURF algorithm. 54

CHAPTER 1

Introduction

The exponential evolution of technology and the fast development of robotic applications has made possible to create autonomous robots that can assist or even replace humans in daily routines and monotonous jobs. Besides their importance in industry, robots begin to enter in domestic environments with the objective of providing services and entertainment to people.

Service robots are autonomous and mobile agents that can assist humans to perform daily tasks in domestic environments. With the development of service robots it is expected that robots and humans can coexist in the same environment, wherein they can provide support and assistance in houseworks and caring of elderly or disabled people. While traditional industrial robots perform their tasks in clearly structured environments with external safeguards, service robots usually work in unstructured environments and collaborate directly with humans. To achieve this, some characteristics must be inherent to the robots. Firstly, it should be able to perceive and build an internal representation for arbitrary home environments, where it must be able to localize itself. Secondly, the robot has to plan itself what to do under different scenarios. In most of the cases, the robot has to be able to navigate through the environment safely, which demands the ability for path planning and obstacle avoidance. At last, the robot is expected to be able of interacting and understanding commands from humans through various methods, namely using natural language.

Sensors give to robots the opportunity to understand and interact with the surrounding environment. Computer vision systems use one of the most rich type of sensors, the cameras, to provide the capacity to perceive and classify the physical world, like people or objects. Such systems must be efficient, since the analysis usually is done in a real time manner. Figure 1.1 shows an example of perception of the world using computer vision.

Computer vision is a topic of study and research in several areas. Currently, vision systems are used in industrial environments, medicine, astronomy, forensic studies, biometrics, among others. One of the main goals in computer vision is the development of a vision system capable of imitate the human vision. The human vision is a complex system that acquire images through the eyes and process them in the brain. In computer vision, the images are

acquired by a camera and processed by a computational unit. All the systems have the same goal of being capable of reacting to external stimulations.

This dissertation focus the use of vision systems for robotic applications, namely in the task of object detection and recognition in domestic environments.

Figure 1.1: Point cloud captured by a robot: example of what a robot can perceive from the world using vision systems.

1.1 OBJECTIVES

The objective of this work is to study and implement object detection algorithms in digital images (2D images) and evaluate their application in mobile robots. Namely, it is intended to develop a computer vision system that can be used on a service robot for the recognition of everyday objects.

The first step is the study of object detection algorithms and some of the previously developed vision systems for service robots. This will help to support the solution design.

After that, a computer vision system, capable of detecting and recognizing objects using some of the studied algorithms, should be developed and integrated in the CAMBADA@Home robot from University of Aveiro. For that, the Kinect camera installed on the top of the CAMBADA@Home robot must be used. The current system of the CAMBADA@Home robot is being developed using the Robot Operating System (ROS) [1], therefore to achieve a better compatibility, and allow for further development, the support for this architecture is important. The system should be tested in order to evaluate the performance of the proposed system.

Finally, the writing of this document should reflect and describe all the work developed.

1.2 THESIS STRUCTURE

The remaining of the document is structured as follows:

- Chapter 2 presents, in the first section, the concept of autonomous and mobile service robots. The second section presents the RoboCup@Home competition and some of the most important teams that have participated in the last editions. The final section of this chapter, describes the Cambada@Home project developed in the University of Aveiro.

- The proposed vision system uses the Microsoft Kinect installed on the top of the CAMBADA@Home robot for image acquisition and it has three main modules. The first module is the responsible to segment a horizontal plane (for example, tables, etc) and detect the objects placed on it. It uses the depth information captured by the camera, in form of a point cloud. It is in the Chapter 3 that this process is described.

- As this work will only explore 2D classification algorithms, a second module described in the Chapter 4, is needded to perform a 2D image projection of each found object. This module acts as a bridge between the object detection module and the object recognition process.

- The last developed module is the responsible of recognizing objects. Some classification algorithms were be studied and implemented in the proposed vision system and they are described in the Chapter 5.

- Chapter 6 explores a test scenario in order to evaluate and validate the developed vision system. The results focus in the overall performance of the proposed vision system using two different classification algorithms.

- Chapter 7 presents the conclusion of this thesis as well the ideas for future work in order to improve the proposed solution.

CHAPTER 2

Object Detection in Service Robots

The continuous growth in robotics technology has played an important role in the evolution of many fields in science and technology. Besides their significant role in industry fields, robots begin to enter in domestic environments to provide services and entertainment to people. With the development of service robots it is expected that robots and humans can coexist, wherein they can provide support and assistance in houseworks and caring of elderly or disabled people.

2.1 service robots

Service robots are agents that can assist humans to perform daily tasks in domestic environments. To achieve this, a service robot has to be capable of avoiding obstacles while navigating in known and unknown environments, recognizing and manipulating objects and understanding commands from humans.

A large number of this type of robots have been developed over last decades by academies and research groups. The results obtained through the conducted experiences will undoubtedly shape the robots of tomorrow in fields such as Face Recognition, Speech Recognition, Sensor Fusion, Navigation, Manipulation, Artificial Intelligence and Human-Robot Interaction.

To be able to interact with the environment [2], the robots can use a variety of sensors. Sensors give to robots the opportunity to understand and interact with the surrounding environment. Computer vision systems use cameras to provide the capacity to perceive and classify the physical world. Such systems must be efficient, since the analysis usually is done in a real time manner.

Nowadays, there are very different types of cameras available [3]. The most used are the RGB cameras (or digital cameras). This type of cameras capture the light variations and create color images. Another type are the thermal cameras. These are able to obtain a completely passive capture of the world based on thermal emissions. The depth cameras (such as *Time-Of-Flight* and stero vision) are capable of perceiving the physical world. From these cameras the captures can be in three forms:

- 2D range data: retrieve a single line of points with two dimensions;
- point clouds: instead of retrieving just one line, it retrieves a set of points in three dimensions;
- depth images: embeds depth information on pixels that compose the image. The intensity of the pixels indicates the distance to the camera.

The last category of cameras that will be listed are the RGB-D cameras that fuses two types of information that complement each other: the color and depth information. An example of this type of cameras is the Microsoft Kinect camera.

Figure 2.1 shows examples of images captured from different types of cameras.

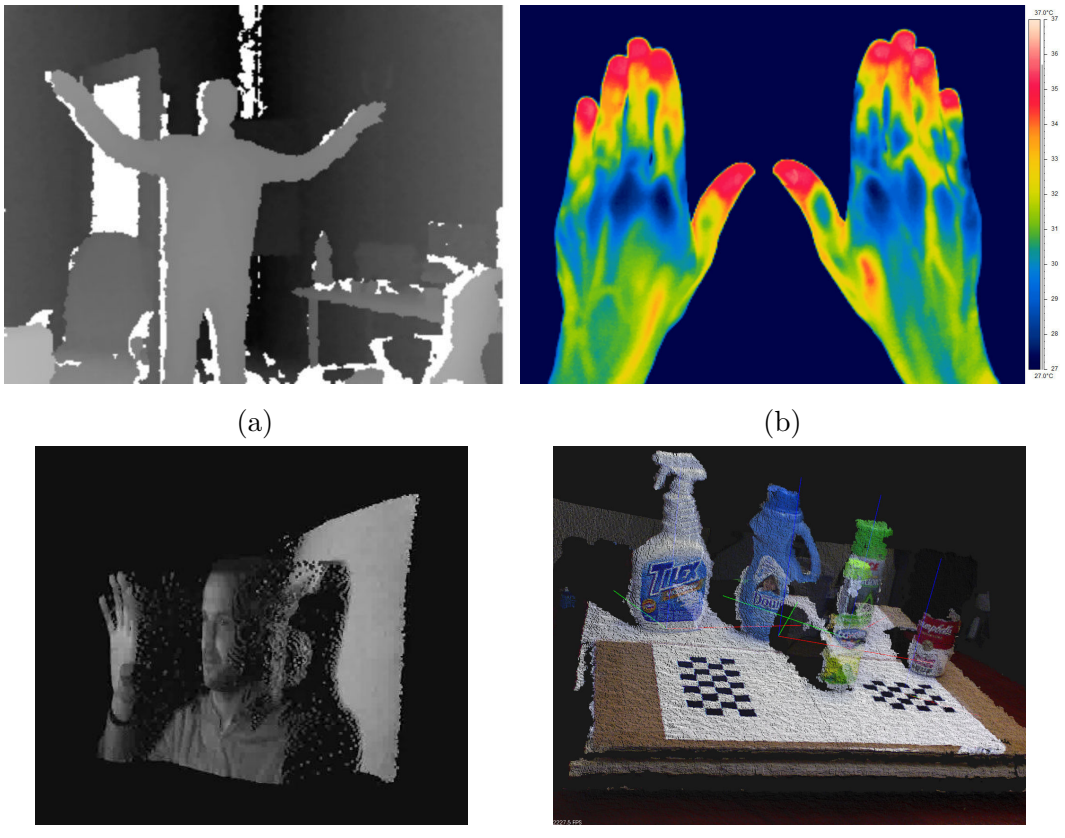


Figure 2.1: Examples of images from different types of cameras found on web: (a) depth image obtained from a depth camera; (b) image captured by a thermal camera; (c) point cloud captured by a *Time-Of-Flight* camera; (d) registered point cloud obtained from a RGB-D camera.

The recognition of objects is fundamental in service robots, mainly for manipulation tasks. This type of vision system requires real-time operation with limited computational resources and it has constraints derived from observational conditions such as the robot geometry, the limited camera resolution and the sensor/object relative pose. According to [4], some of recent approaches used in service robots (especially the ones used in RoboCup@Home league) are structured so that first detects horizontal surfaces (for example, a table or the ground) for restricting the search area of the possible object's positions, and then compute features in order to recognize the objects. Popular features include the use of visual appearance, calculating local interest points (keypoints) and descriptors (e.g., SIFT [5] and SURF [6]) and/or the use of 3D feature descriptors such as feature histograms obtained from range images. In most of the cases, the robotic platforms are equipped with RGB and/or RGB-D cameras for the data acquisition.

2.2 ROBOCUP@HOME

RoboCup [1] is an annual international competition that aims the developing of artificial intelligence, robotics, and related fields. It is an attempt to motivate artificial intelligence and intelligent robotics research by providing challenges where the new developments in these areas can be tested and examined.

The RoboCup@Home [2] league aims to develop service and assistive robot technology with high relevance for future personal domestic applications. It is part of the Robocup competition and is the largest annual competition for autonomous service robots. A set of benchmark tests is used to evaluate the robots abilities and performance in a realistic home environment setting. It focus on several domains as Human-Robot-Interaction and Cooperation, Navigation and Mapping in dynamic environments, Computer Vision and Object Recognition under natural light conditions, Object Manipulation and Adaptive Behaviors.

The following are some of the teams that participate in the RoboCup@Home and a brief description of their vision systems (the descriptions are taken from the team descriptions papers submitted by each of the teams to the robocup@home competition):

- **WrightEagle** is a RoboCup team created by the Multi-Agent Systems Lab. of University of Science and Technology of China. This team has been developing a robot called KeJia [7] (see Figure 2.2). Their vision system consist of a Microsoft Kinect camera and a high-resolution RGB camera from PointGrey [3]. Their system obtains an aligned RGB-D image by combining the RGB image with the depth image from Kinect and with such aligned RGB-D image, the vision module is capable of detecting, tracking people and recognizing different kinds of objects. They follow the approach as proposed in [8]

[1] http://www.robocup.org/
[2] http://www.robocupathome.org/
[3] http://www.ptgrey.com/

to detect and localize table-top objects including bottles, cups, etc. The depth image is first transformed and segmented, then the largest horizontal plane is extracted using Point Cloud Library (PCL) [9], and point clouds above it are clustered into different pieces. After that the SURF feature matching against the stored features are applied to each piece. The one with highest match above certain threshold is considered as a recognition. At last, to further enhance the detection performance and decrease false positive rate, they check each recognized cluster and filter out those vary too much in size.

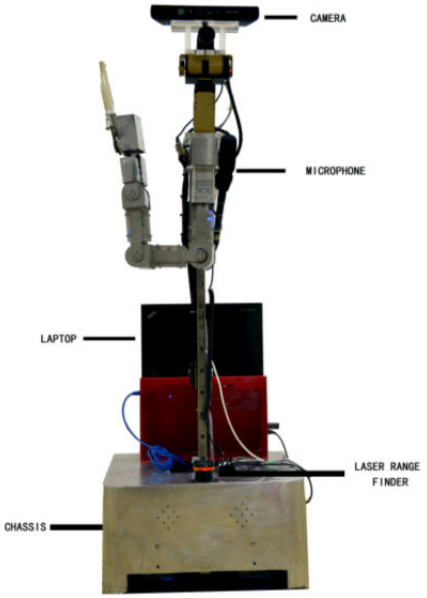


Figure 2.2: The KeJia robot.

- **b-it-bots** RoboCup@Home team at Bonn-Rhein-Sieg University of Applied Sciences (BRSU), Germany, has been established in 2007 [10]. Until 2011, they used the robot plataform "Johnny Jackanapes", a differential drive robot based on the Volksbot platform. Nowadays, they changed to "Jenny", the omni wheeled robot Care-O-bot 3 (see Figure 2.3). The Care-O-bot 3 is developed by the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) in Stuttgart, Germany [4]. The sensor head of the robot contains a zoomable Stemmer CVC EH6500 HD GE/POE color camera and a Microsoft Kinect camera. In order to detect objects in cluttered domestic environments, they proposed an object detection method that copes with multiple plane and object occurrences like in cluttered scenes with shelves. Further a surface reconstruction method based on Growing Neural Gas (GNG) [11] in combination with a shape distribution-based descriptor was proposed to reflect shape characteristics of object candidates. Properties provided by the GNG such as smoothing and denoising effects support a stable description of the object candidates which also leads towards

[4]http://www.care-o-bot.de/en/research.html

a more stable learning of categories. Based on this descriptor a dictionary approach combined with a supervised shape learner was presented to learn prediction models of shape categories.

Figure 2.3: The Care-O-bot robot.

- **Tech United Eindhoven** is the RoboCup team of the Eindhoven University of Technology, competing in the Middle Size League and the @Home League. Tech United has been competing in the @Home league since 2011 [12]. They have competed in the RoboCup@Home with the AMIGO robot (see Figure 2.4), that uses a Microsoft Kinect camera for object detection and recognition. Its vision system has a pipeline containing a sequence of different algorithms. Under normal operation, the first component is an object segmentation module. Objects on top of a horizontal plane are segmented using the point cloud library (PCL). This module also approximates the size of the segmented objects and based on the size selects a subset of candidate objects from the object database. The segmented objects and the related color image regions are given to the next module which contains a custom implementation of the linemod algorithm [13]. This algorithm recognizes objects based on 2D and 3D gradient templates and color information. The last module is based on the objects of daily use finder perception system [14], which is implemented by the ROS ODUfinder package[5]. This system aims at recognizing textures objects by adopting SIFT features using vocabulary trees. The object detection and recognition system has a probabilistic nature. Each of the candidate objects given to the next module is associated with a probability. The module

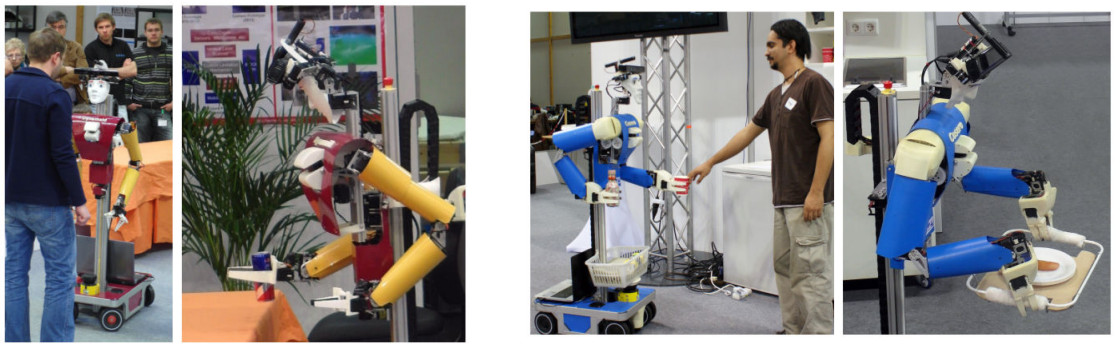
[5]http://wiki.ros.org/objects_of_daily_use_finder

refines the probabilities and removes unlikely object candidates. It provides the object position together with a probability mass function over the possible class labels to the world model. If none of the objects is associated with a sufficient probability, the world model will insert an "unknown" object at the given location.

Figure 2.4: The AMIGO robot.

- **NimbRo** is the RoboCup team of Rheinische Friedrich-Wilhelms-Universit at Bonn, Germany [15]. This team has been developing two robots, Dynamaid and Cosero (see Figure 2.5). The robots perceive their environment with a variety of sensors. For detection of small obstacles on the floor, a Hokuyo URG-04LX LRF is mounted between the front wheels. For 3D perception, a tilting Hokuyo UTM-30LX LRF (laser range finder) is mounted in the chest. For measuring the height and distance of support surfaces, e.g. table tops, and detecting objects on these, a URG-04LX LRF is mounted on a roll joint on the hip. The head contains a RGB-D camera (Microsoft Kinect camera) and a directed microphone. Both can be directed at objects or persons by a pan-tilt mechanism in the neck. To detect the presence of objects, the grippers are equipped with IR distance sensors. Recently, Cosero has been equipped with an additional camera in the belly. They had developed efficient RGB-D SLAM (Simultaneous Localization and Mapping) methods, based on Multi-resolution Surfel Maps (MRSMap) [16], which run in real time on a CPU. These can be used to model the environment and localize in these maps, or to obtain 3D object models from multiple views and track these in the camera images. In addition to recognition of known object instances, which is based on SURF features and color histograms, they also developed methods for 3D semantic mapping, which are based on RGB-D SLAM and random forest object-class

segmentation [17].



(a) (b)

Figure 2.5: NimbRo cognitive service robots: (a) Dynamaid; (b) and Cosero.

2.3 CAMBADA@HOME

The development of a robotic platform for elderly care in the University of Aveiro started with the project named Living Usability Lab for Next Generation Networks. The project was a collaborative effort between the industry and the academy that aims to develop and test technologies and services that give elderly people a quality lifestyle in their own homes while remaining active and independent [18].

Under the Living Usability Lab Project, of which DETI/UA (Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro) was a partner and with the past experience obtained with the project CARL (Communication, Action, Reasoning and Learning in Robotics) [19], the goal was the development of an assistant robot. This robot must move in a residential space, having to interact with its environment and the humans who use the same space. DETI/UA already has significant experience in projecting and developing robots. In particular, its robotic soccer team CAMBADA [6] is constituted by robots which, because of their characteristics, may be used as a base for the assistant robot project, although the environment in which they operate, a football pitch, is quite different from a residential space.

The CAMBADA@Home [20] is the University of Aveiro RoboCup@Home team. The project was created in January 2011 following the team past experience in the CAMBADA robotic soccer team. The development of the CAMBADA soccer team started in 2003 and a steady progress was observed since then. CAMBADA has participated in several national and international competitions, including RoboCup world championships, the European RoboLudens, German Open and the annual Portuguese Open Robotics Festival.

[6] CAMBADA is an acronym for Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture.

The CAMBADA@Home platform is designed as a three layer mechanical/electronic platform which can accommodate in an effective way the number of sensors and actuators needed to perform the RoboCup@Home challenges. The vision system is located on top of robot, using a Kinect sensor. In Figure 2.6 it is possible see the current platform of the CAMBADA@Home robot.

Bottom layer adopts a four motor system which drives a symmetrical set of omni-wheels ensuring a stable and versatile motion solution. This layer also includes three Li-Po batteries, all the low level control hardware, a SICK LMS100 laser find ranger (LRF) and the support for a vertical linear actuator. It also allows a standard laptop to be located in the back side and has room for other needed equipment such as a switch for a robot local network. The second layer represents the torso of the robot and can be moved up and down by means of the linear actuator, allowing the size of the robot to be continuously adjusted between 0.95m and 1.40m high. On top of the torso, a third layer is developed upon a pan and tilt structure that holds a Kinect plus a thermal vision system. This layer symbolically represents the head of the robot. Manipulators are crucial to provide the robot with the capability to perform manipulation tasks such as grabbing objects, openning bottles, manipulating device interfaces, among others. The development of a set of two anthropomorphic arms, each with 7 degrees of freedom (DOF), aims to execute the above mentioned tasks, contributing to the evolution of current version of CAMBADA@Home robot [20]. Figure 2.7 shows the desired final platform design.

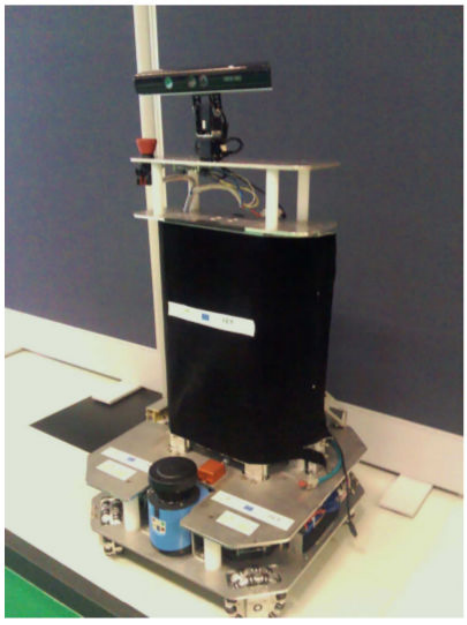


Figure 2.6: CAMBADA@Home: current platform.

The CAMBADA@Home robot is equipped with a LRF for the purpose of mapping and obstacle detection. Besides the use of the LRF, also the Kinect camera is used to detect the nearest obstacles. To perceive the world that surrounds the robot, a thermal camera and the Kinect camera are used with the purpose of people and objects detection.

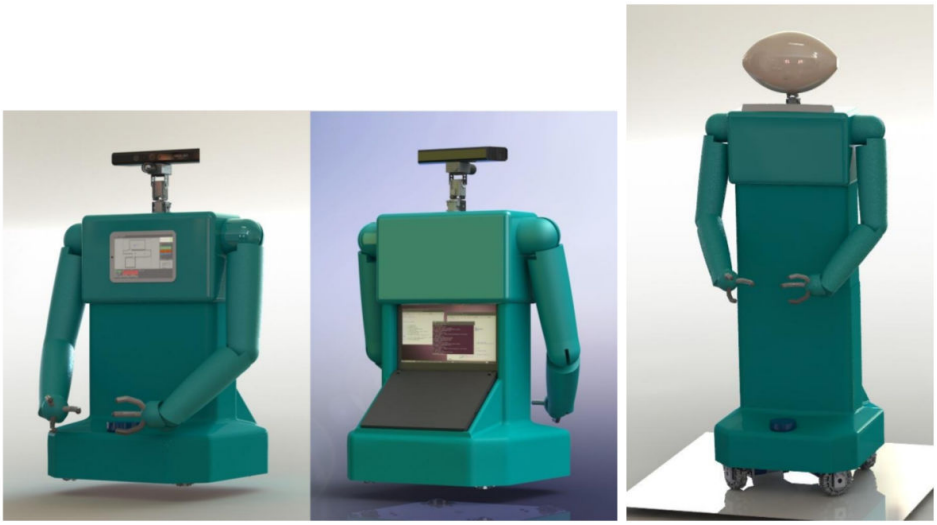


Figure 2.7: CAMBADA@Home: final platform design.

The existing vision system of the current platform has modules capable of detecting and tracking humans, allowing it to interact with the people around it, but do not have yet the capability to detect objects. In order to complete many of the challenges presented in the RoboCup@Home competition, a vision system that achieves this ability must be developed and it is in this field that the work described in this document is focused. This project was the first attempt to approach the problem and it will act as a basis for future developments.

CHAPTER 3

Depth Segmentation

The proposed vision system uses the Microsoft Kinect installed on the CAMBADA@Home robot for image acquisition and it has several modules. The first module is the responsible for detecting objects placed on a horizontal plane, like the ground or a table, being this module described in this chapter. Later in this document will be explained the other two modules which aim at object recognition.

This chapter presents the external frameworks and libraries used in the development of the proposed vision system. Moreover, in this chapter is explained the work performed by the system to process the depth information of the scene (in a form of a point cloud) in order to segment and extract objects placed on a horizontal plane - object detection algorithm.

3.1 External Software

The system was developed using C++ Programming Language. As all the previous work on the CAMBADA@Home robot is based on the ROS framework, also this system has to be compatible with it. PCL library is used for the processing of the point clouds and OpenCV for 2D processing and classification. Next it will be explained what these frameworks are and how these are explored by the system developed.

PCL - Point Cloud Library

A point cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface and when color information is present, the point cloud becomes 4D. Point clouds can be created by using, for example, stereo cameras, 3D scanners, time-of-flight cameras, or synthetically generated using a computer program.

The Point Cloud Library (PCL) [9] is a standalone, large scale, open project for 2D/3D image and point cloud processing. The PCL framework contains a set of state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, segment relevant parts of a scene, create surfaces from point clouds and visualization (see an example of application of the PCL in Figure 3.1).

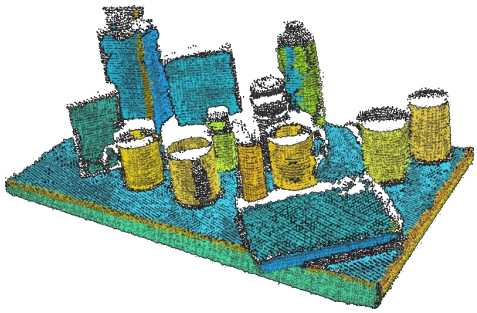


Figure 3.1: Point Cloud Library example: application of SAmple Consensus (SAC) methods like RANSAC and models like planes and cylinders.

OPENCV - OPEN SOURCE COMPUTER VISION

The OpenCV library is a computer vision library [21], initially launched by *Intel* and later supported by *Willow Garage*. Currently, it is maintained by *Itseez* [1].

OpenCV focuses mainly in real-time image processing and computer vision applications (see an example in Figure 3.2) and it provides several algorithms, spanning from many very basic tasks (capture and pre-processing of image data) to high-level algorithms (feature extraction, machine learning, motion tracking). It has C++, C, Python and Java interfaces and it is plataform independent.

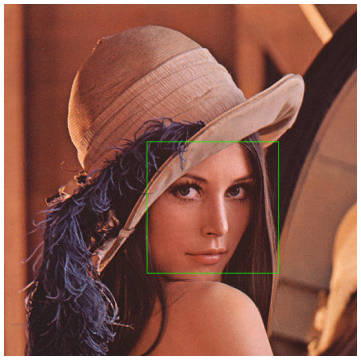


Figure 3.2: OpenCV Library example: face detection using OpenCV.

[1] http://itseez.com/

ROS - ROBOT OPERATING SYSTEM

The Robot Operating System (ROS) [1] [14] is a flexible framework for the development of software intended to be used with robots. It is a collection of tools, libraries, and conventions that aim to simplify the process of developing complex and robust robot behavior across a wide variety of robotic platforms. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more.

ROS is a peer-to-peer network of processes that are coupled using the ROS communication infrastructure. ROS supports and implements several different styles of communication, including synchronous RPC[2]-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. A node can be defined as an executable file within a ROS package. ROS nodes use a ROS client library to communicate with other nodes. Nodes can comunicate with each other by publishing or subscribing to a Topic. Messages are ROS data type used when subscribing or publishing to a topic - Figure 3.3.

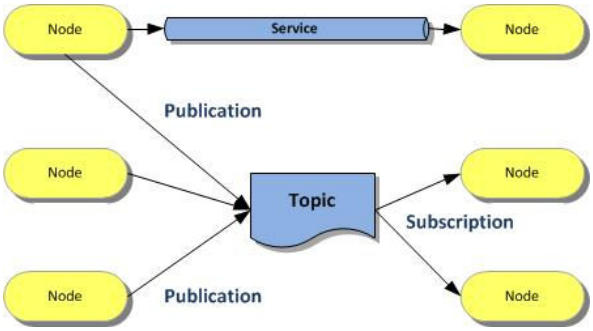


Figure 3.3: Robot Operating System: communication diagram example.

To access the Kinect data, the system subscribes a topic published from the Kinect ROS driver node. The driver used is the freenect [3], because it provides a registered point cloud, i.e., a point cloud that represent the spatial information of a scene and each point is associated with the correspondent color information.

3.2 PROPOSED APPROACH

Before the development of this system, it was defined an heuristic considering that only objects that are placed on a plane, such as on a table or on the ground, should be considered. Therefore, the developed vision system should be able to identify and segment horizontal planes. With this assumption, two physical conditions are imposed to the robot: the Kinect must be slightly inclined towards the ground direction and it should be positioned above the height of the plane, in order to guarantee that the plane can be correctly seen - Figure 3.4.

[2]Remote procedure call
[3]https://github.com/OpenKinect/libfreenect

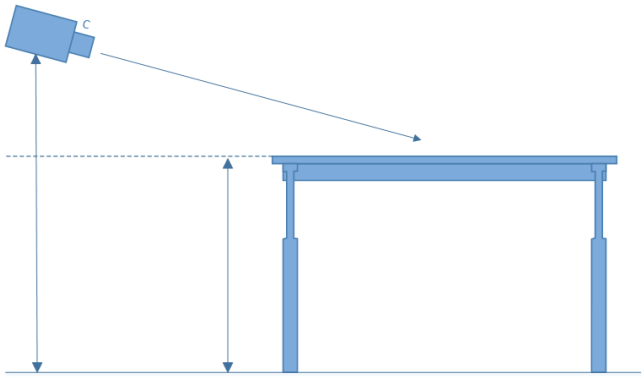


Figure 3.4: Kinect position and orientation relative to a plane.

As stated before, the application subscribes a topic from the ROS kinect driver (freenect). In the first message received from the topic, i.e. in the first iteration, the system performs the calculation of the rotation vectors that will be used in the pre-processing of the point cloud provided.

The first step is the pre-processing of the received point cloud. After that, the result of this process is passed to the segmentation step. If the segmentation is correctly performed, a process of extraction of the objects lying on the plane starts. In the end, a set of objects is given to the next part of the developed system that is only described in the Chapter 4.

Figure 3.5 shows the workflow performed by the system in order to segment a plane and extract the objects placed on it. Initially, the captured point cloud is filtered in order to reduce the number of points to be processed. Next, if a horizontal plane is present in the scene, it will be segmented and the objects lying on it will be extracted.

Each component of this diagram will be presented and discussed in more detail on the next sections.

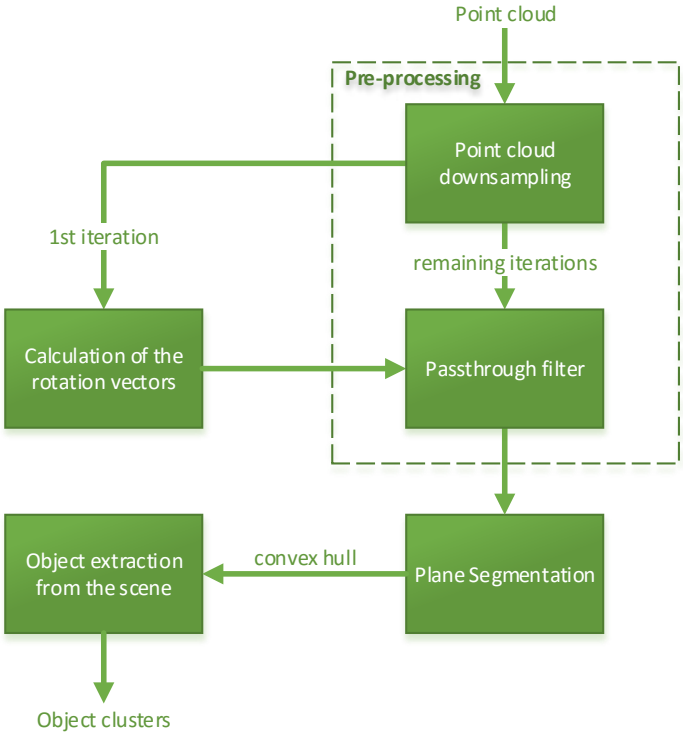


Figure 3.5: Depth segmentation: structure of the proposed approach.

Figure 3.6 shows an example of an original point cloud provided by the freenect driver without any type of pre-processing applied by the system.

The rotation vectors are calculated in the first iteration of the execution of the system. This vectors are crucial because for the correct application of the passthrough filter (filter that removes points outside a given range), the point cloud must be aligned with the XZ plane, taking as reference the dominant horizontal plane of the scene.

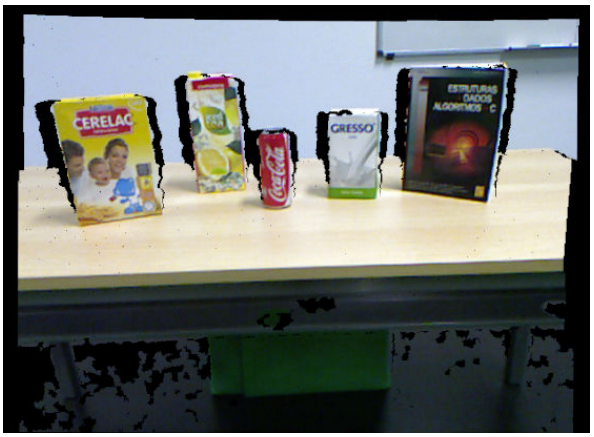


Figure 3.6: Screenshot of a point cloud provided by the Kinect ROS driver.

3.3 CLOUD PRE-PROCESSING

The point cloud provided by the Kinect ROS driver comes in the form of an organized 2D image of 640 rows by 480 columns, resulting in a total of 307200 three dimensional points. This section describes two implemented filtering processes with the objective of reducing the number of points and consequently the processing time, resulting in a higher efficiency of the system.

The first filtering process applied to the input data is a voxel grid filter, provided by the PCL library. The purpose of this filter is the downsampling of the 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 (see Figure 3.7) over the input point cloud data. Then, in each voxel (i.e., 3D box), all the points inside it will be approximated (i.e., downsampled) with their centroid.

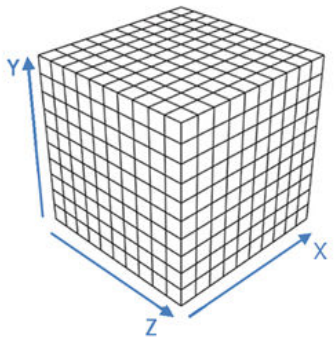


Figure 3.7: 3D voxel grid.

In order to use the PCL implementation of this filter, an input parameter, called *leaf_size*, has to be defined. This parameter is the side dimension of each voxel. In Table 3.1 there are some experimental results obtained during the study of the value of this parameter.

Leaf Size (m)	**Before Filter (points)**	**After Filter (points)**	**Processing Time**
0.002	307200	208007	23.138 ms
0.004	307200	104835	21.008 ms
0.006	307200	56913	19.585 ms
0.008	307200	34571	18.285 ms
0.01	307200	22751	17.423 ms
0.012	307200	16098	16.043 ms
0.014	307200	11952	15.482 ms
0.016	307200	9247	15.171 ms
0.018	307200	7361	14.923 ms
0.02	307200	5984	14.221 ms

Table 3.1: Voxel Grid Filter: efficiency results.

In Table 3.1 it is possible to observe that the increasing of the *leaf_size* value results in the reduction of the point cloud size (i.e., the number of points) and the processing time of this filter.

During the experimental tests, it was observed that *leaf_size* values greater than 0.006 meters (6 millimeters) can eliminate some parts of the point cloud structure of the scene and compromise the correct execution of the developed vision system.

Also note that, the resulting point cloud after the aplication of the voxel grid filter, using a *leaf_size* = 0.006 meters, has about 18.5% points from the original point cloud.

After the application of the voxel grid filter, a second filtering process is applied. The objective of this filter is to remove all the points outside the defined height limits. These limits will define the height at which it is expected that the horizontal plane is placed. To do that, another function from PCL is used: the passthrough filter. This function iterates through the entire input point cloud once, filtering and removing all the points outside the specified interval, which applies only to the specified field. In this case, the interval is defined by the user before the execution of the vision application and the field is the height (i.e., in Y-axis).

As this filter acts through a defined axis and the input point cloud is not aligned with the XZ plane, the cloud must be oriented and aligned using the rotation vectors calculted in the first iteration. Only after this, the passthrough filter can be applied.

To align the cloud, four transformations are executed: three rotations (one per axis) and one translation in the Y-axis. The purpose of the last transformation is to translate the cloud, so that the minimum point on the Y-axis coincides in $Y = 0$. This requires that the robot must see in every iteration a portion (even if small) of the ground.

In the end, the filter is applied (removing outer points) and the pre-processed point cloud is aligned again to its original position.

The Figure 3.8 shows an example of a point cloud after application of the voxel grid filter and the pre-processing step.

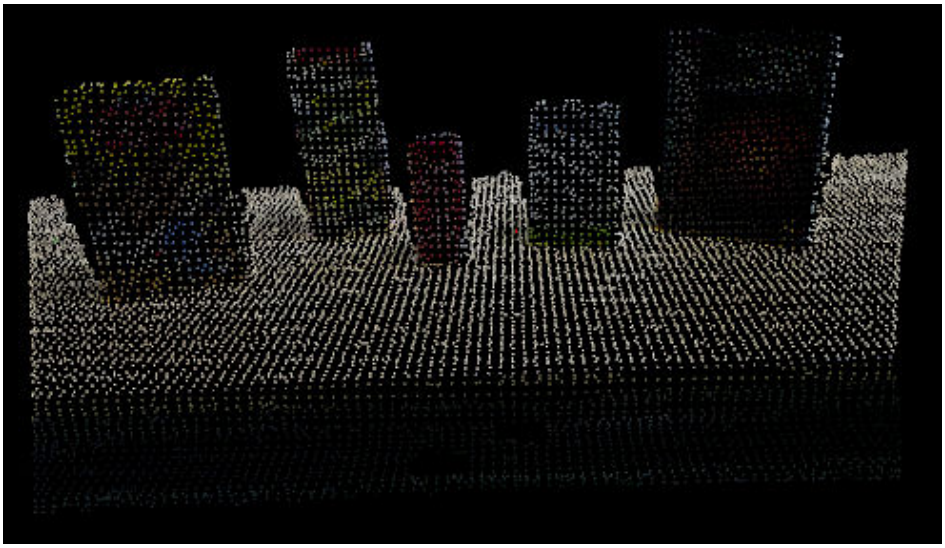


Figure 3.8: Example of a pre-processed point cloud after the application of the two described filters; *leaf_size* = 0.006 meters (6mm). The original point cloud was presented in Figure 3.6.

3.4 PLANE SEGMENTATION

After the pre-processing of the provided point cloud, the system tries to segment the dominant plane presented in the scene. PCL has functions that try to find all the points of an input cloud that fit a plane model. This system uses one of these functions that uses an iterative method known as RANSAC (RANdom SAmple Consensus) [22].

The RANSAC algorithm is widely used in the field of visual computing. This algorithm allows the estimation of parameters of a model in a given dataset. It works under the assumption that the data contains inliers (data can be adjusted to the model, even with a little noise) and outliers (data that does not fit the model at all).

In each iteration, a small number of points are randomly selected. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. Next, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above steps until the obtained consensus set in certain iteration has enough inliers.

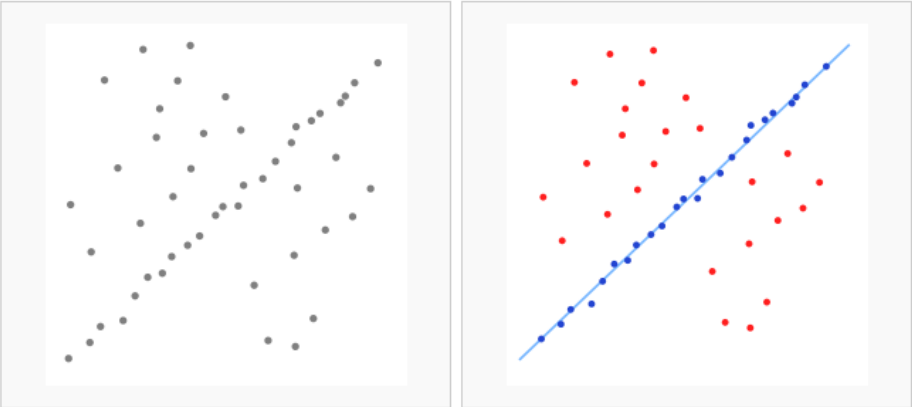


Figure 3.9: Fitted line with RANSAC; outliers have no influence on the result.

Beyond the input cloud data, the function execution depends on two input parameters:

- Distance Threshold: This parameter defines the distance from the estimated parametric model. All the points that are inside this range are considered inliers, ie., they belong to the plane.
- Max Iterations: Maximum number of times that the RANSAC will iterate. On every iteration, the accuracy of the result improves, but it will influence the processing time.

In the proposed system, this function is configured to segment a plane perpendicular to Y-axis. This mean that it is only capable of segmenting horizontal planes, that is a predefined heuristic of the developed system.

A set of indices are given as the segmentation results. These indices are then used to extract the segmented plane from the given point cloud and all the points belonging to it are stored in an auxiliar point cloud. After the extraction, a convex hull from the plane is calculated. A convex hull of a given set of points in the plane is the smallest convex polygon that contains all the points of it (see Figure 3.10).

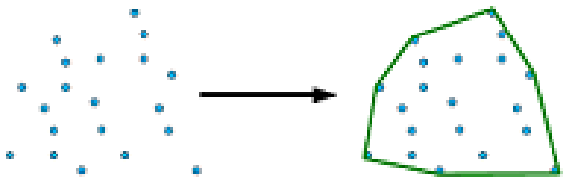
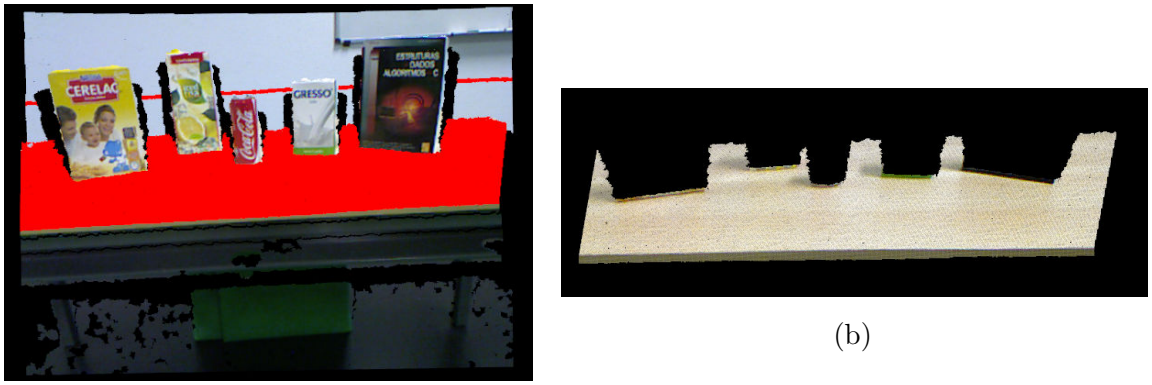


Figure 3.10: An example of a convex hull of a set of points.

The Figure 3.11 shows an example of a plane segmentation.



(a) (b)

Figure 3.11: Plane segmentation example: (a) example of a segmented plane in red; (b) extracted plane resulting from a segmentation.

3.5 OBJECT EXTRACTION FROM THE PLANE

After the process of plane segmentation, the system has as results an auxiliary point cloud with all the points inherent to the segmented plane and the correspondent convex hull. It is considered that all the points lying on the segmented plane belong to graspable objects. With this information, the next step is to filter the point cloud resulting from the application of voxel grid filter, in order to get only the points that satisfy the previous consideration.

It is used a function, from the PCL library, to extract all the points between a certain range from the plane, with their projections falling inside the given convex hull. As the purpose is to detect household objects (which typically have a small size), the defined interval is between 0.02 and 0.5 meters, what means that the objects must not have a height exceeding 0.5 meters. The value 0.02 is to cut out the segmented plane and only retrieve the points above this plane. The result is a set of points that belong to the objects.

Given this result, a clustering algorithm has to be applied to identify individual objects. The clustering is essential to subdivide the point cloud in subsets of points (clusters) that possibly maps a distinct object.

A simple function is present in PCL library. In order to reduce the overall processing time, this method needs to divide the given point cloud into smaller parts before the clustering. A simple approach can be implemented by making use of a 3D grid subdivision. For this, a Kd-tree representation of the input point cloud has to be created. A Kd-tree is a binary search tree that stores points in k-dimensional space. It is useful when it is necessary to use searches involving a multidimensional search key, like in k-nearest neighbor searches (that is used in this function).

Next, it is presented the pseudocode [23] of the clustering algorithm used and present in the PCL library..

Algorithm 1 EuclideanClusterExtraction

create a Kd-tree representation for the input point cloud dataset P
set up an empty list of clusters C, and a queue of the points that need to be checked Q
for every point $p_i \in P$ **do**
    add p_i to the current queue Q
    for every point $p_i \in Q$ **do**
        search for the set P_k^i of point neighbors of p_i in a sphere with radius $r < dist_{th}$;
    end for
    when the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list
end for
the algorithm terminates when all points $p_i \in P$ have been processed and are now part of the list of point clusters C

The result of this process is a set of point clouds, each one representing an individual object cluster (see Figure 3.12).

Figure 3.12: Screenshot of the set of point clouds resulting from the clustering process.

3.6 RESULTS

To validate the depth segmentation process of the developed vision system, some experiments were conducted with the CAMBADA@Home robot. The processing unit of the robot is an Intel Core i5-3210M CPU @ 2.50GHz 4 processor, running Linux (distribution Ubuntu 14.04. LTS Trusty Tahr).

The scenario chosen for the experiments was a common table and the objects were items usually found in household environments. Some tests were performed and they only differ in the number of objects lying on the table. The results presented in this section only take into account the work described until this point.

For all the scenarios, the robot moved in (closer to the table) and then moved away, in a straight line. The following parameters were used:

- Pre-processing: *leaf_size* = 0.006 meters (6mm); *minHeight* = 0.5 meters (50cm) and *maxHeight* = 1 meters;
- Object extraction: *distance_threshold* = 0.01 meters (1cm).

The purpose of these experimental tests was to measure the efficiency (results presented in Table 3.2 and Table 3.3) and the accuracy (Table 3.4) of the depth segmentation process, namely the processing time and the average number of found clusters.

Scenario	**Plane alignment time**	**Total time**
1 object	16.264 ms	58.661 ms
2 objects	17.984 ms	62.902 ms
3 objects	17.251 ms	66.152 ms
4 objects	17.949 ms	72.110 ms
5 objects	17.105 ms	72.651 ms

Table 3.2: Experimental results: processing time of the calculation of rotation vectors in the first frame.

Scenario	**Voxel grid time**	**Pre-processing time**	**Segmentation time**	**Clustering time**	**Total Time**
1 object	23.430 ms	2.323 ms	1.986 ms	9.082 ms	38.012 ms
2 objects	23.668 ms	2.836 ms	2.152 ms	10.365 ms	40.505 ms
3 objects	23.863 ms	2.731 ms	2.313 ms	11.191 ms	41.735 ms
4 objects	23.854 ms	2.543 ms	2.564 ms	11.762 ms	42.177 ms
5 objects	23.993 ms	2.702 ms	2.365 ms	12.473 ms	43.091 ms

Table 3.3: Experimental results: efficiency of the developed system; 150 frames were analysed.

The rotation vectors are calculated in the first frame (i.e., first iteration) of the system execution. For this reason, this process only affects the processing time in the first frame. In Table 3.2 are presented the processing times of this process for all the tested scenarios. It is possible to observe that the time is similar in all cases. This can be explained by the fact that the table used was the same for all tests as well as the orientation of the Kinect installed on the robot. Another observation is that the number of objects does not affect the processing time of the calculation of the vectors.

In Table 3.3 are presented the processing times of all the processes described in this chapter. In this table, it is possible to see that the process that takes longer to execute is the

voxel grid filter. This filter is invariant to the number of objects present in the scene as also the pre-processing and segmentation processes.

As can be seen in Table 3.3, the clustering processing time is higher as the number of objects above the table, which was expected, since the greater the number of objects, greater is the number of points that must be analyzed and processed by clustering step.

The overall processing time is then dependent of the number of objects present above the table. In Figure 3.13 it is presented some figures illustrating the retrieved clusters for each scenario.

Scenario	**Average number of found clusters**	**Detection Ratio**
1 object	1.02	98 %
2 objects	2	100 %
3 objects	3	100 %
4 objects	4.01	98.67 %
5 objects	5	100 %

Table 3.4: Experimental results: Detection ratio of the developed system; 150 frames were analysed.

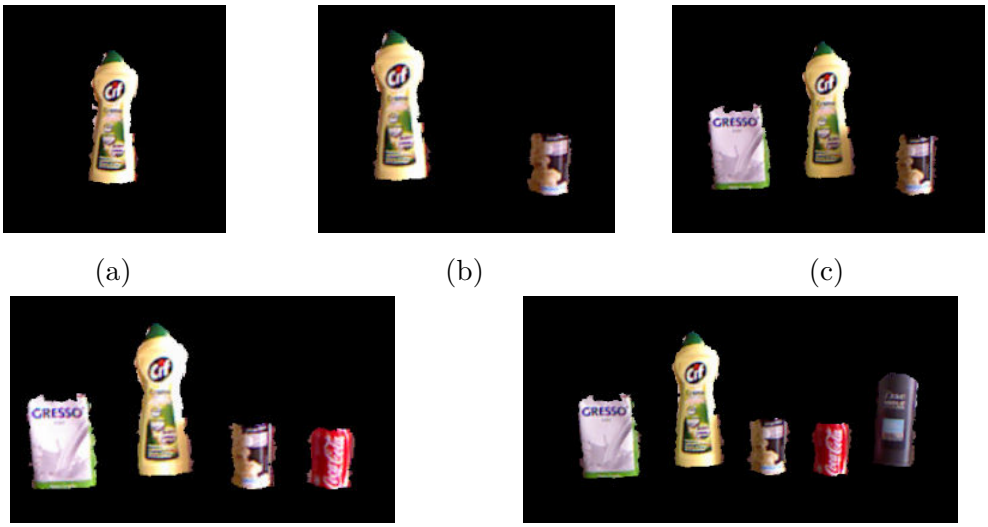


Figure 3.13: Clustering process: retrieved clusters for each scenario.

The Table 3.4 shows the results of the accuracy analysis for each of the test cases. It is important to note that all the tested scenarios are simple controlled environments, with few objects and noise. But for all the cases, the developed vision system presents an detection ratio equal or even higher than 98 % in detecting objects.

CHAPTER 4

Image Segmentation using depth information

The objective of this work is to implement a vision system capable of detecting and recognizing objects on digital images. Chapter 3 describes the algorithms implemented to detect objects using 3D information. This chapter describes the process responsible by the projection of the detected 3D objects into the RGB digital images that will be the input for the object recognition algorithm described in Chapter 5.

4.1 PROPOSED APPROACH

So far, all the processing was performed in the 3D space. As defined in Section 1.1, the developed vision system will only explore 2D classification algorithms and all the objects will be treated as 2D objects. Because of this, each of the detected clusters (potential objects) must be project to the 2D space of the RGB image.

Figure 4.1 illustrates the workflow performed by the system in order to project all the clusters to the 2D space. Each component of this diagram will be presented and discussed in more detail on the next section.

In the Figure 4.1, the blue color component is related to a process that runs only once per iteration. All the other components are executed for each found object in every iteration.

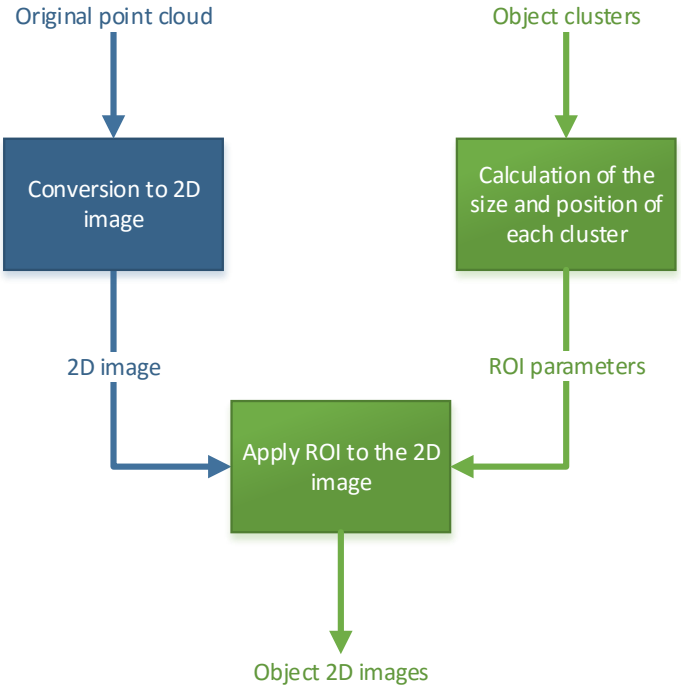


Figure 4.1: Color Segmentation: structure of the proposed approach.

4.2 REGISTRATION PROCESS

An organized point cloud is one that resemble an organized image like structure, where the data is organized as a 2D array of points, with a specific width (total number of points in a row, i.e., total number of columns) and height (total number of rows). An unorganized point cloud might contain the same data of an organized point cloud, also providing all the 3D coordinates of each point, but in this case, the data is organized as a 1D array of points, i.e., the width assumes a value equal to the total number of points and the height is 1.

This process takes as input the set of point clouds of each found cluster resulting from the processes described in the Chapter 3. At this step, the objective is to obtain an individual 2D image of each cluster, instead of using the entire image, in order to be possible to apply object detection algorithms.

A simple solution to form a 2D image of a cluster is to iteratively project all the 3D points to the 2D space. This could be the solution applied if the point cloud of the cluster was an organized point cloud. But, with the application of all the steps described in the Chapter 3, the resulting point clouds became unorganized. This is an implication of using the filters applied in the pre-processing process and the use of the custering/extraction algorithm.

The Kinect ROS driver publish a topic that provides an organized point cloud with RGB information. This point cloud is registered with color information by the driver, which means that a direct correspondence from the 3D space to 2D space can be easily establish.

Then, the chosen approach was the calculation of a region of interest (ROI), in the 2D image of the original point cloud, that fits all the points of a cluster. So, two things are needed: the 2D projection of the original point cloud and the parameters that define a ROI for each cluster.

The first step is the cloud conversion/projection to a 2D image. Following, the pseudocode of the process of form an image from the original point cloud is presented.

Algorithm 2 Conversion of an organized point cloud to a 2D image

1: create an empty cv::Mat structure, img, with the dimensions of the original point cloud ($rows \times cols$);
2: **for** every row $r_i \in rows$ **do**
3: **for** every col $c_j \in cols$ **do**
4: get the 3D point $P_{3D}(c_j, r_i)$ from the point cloud with coordinates (c_j, r_i);
5: get RGB information of the point;
6: add the gathered RGB information to the point $P_{2D}(c_j, r_i)$ of img;
7: **end for**
8: **end for**
9: the algorithm terminates when all points P_{3D} have been processed;

As was illustrated in the previous algorithm, the process begins with the creation of an empty cv::Mat data type provided by the OpenCV library. The size of this structure is the same of the original point cloud, i.e., the same width (number of columns) and height (number of rows). In this case, as the camera used is a Microsoft Kinect, the image size is 640 pixels per 480 pixels. After the creation of the empty image, the proceeding is to iterate all the points of the cloud, extracting the color (RGB) information of each point and adding it to the corresponding pixel in the image. The result (see an example in Figure 4.2) will be a 2D projection of the original point cloud provided by the ROS Kinect driver.

(a) (b)

Figure 4.2: Projection result of the original point cloud to a 2D image: a) original point cloud; b) 2D image projection of the point cloud.

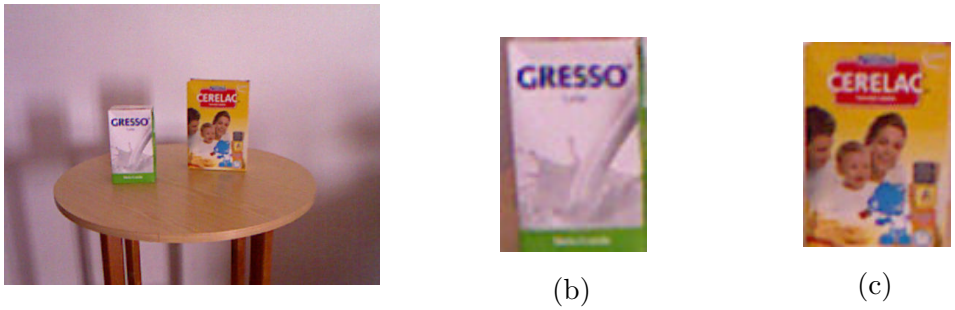
The second step, is the calculation of the size and position of the cluster in the original point cloud, to find the necessary parameters to define a ROI. As the point cloud of each cluster is unorganized and subsampled, a direct correspondence can not be establish. Then, the chosen approach was to iteratively project all the 3D points of the cluster to pixels, in order to find the maximum and minimum values in the 2D space, i.e., the maximum and minimum values to the X and Y coordinates. With a well defined ROI, it is desirable that for each cluster correspondes an image that only englobe the found cluster. To project the points, it is used a function from the PCL library, that takes as input a 3D point and return its 2D projected point.

Following, the pseudocode of the algorithm responsible for calculate the necessary parameters is presented.

Algorithm 3 Calculation of ROI parameters

given an input cluster C, it is defined a minimum 2D point $min\_pt = (640, 480)$ and a maximum 2D point $max\_pt = (0, 0)$;
for every $point_i \in C$ **do**
    project $point_i$, that returns a $pixel_i$;
    if $pixel_i.x < min\_pt.x$ **then**
        $min\_pt.x = pixel_i.x$
    end if
    if $pixel_i.x > max\_pt.x$ **then**
        $max\_pt.x = pixel_i.x$
    end if
    if $pixel_i.y < min\_pt.y$ **then**
        $min\_pt.y = pixel_i.y$
    end if
    if $pixel_i.y > max\_pt.y$ **then**
        $max\_pt.y = pixel_i.y$
    end if
end for
the algorithm terminates when all points $point_i \in C$ have been processed;

Finished the calculation of the ROI parameters of a cluster, the final step is to apply the ROI on the 2D image of the original cloud. This step acts as a filter: it creates a sub image, with the ROI size, where the pixels that fit outside the ROI are discarded, resulting in a 2D projection of a specific cluster (see Figure 4.3).



(a) (b) (c)

Figure 4.3: Resulting images of the objects after the application of the ROI.

4.3 RESULTS

To evaluate the time processing of this module of the developed vision system, some experiments were conducted with the CAMBADA@Home robot. The scenario chosen was the same that is described in the Section 3.6.

Some tests were performed and they only differ in the number of objects lying on the table. The results present in this section only take into account the work described in this section, except the overall processing time ("Total time" in Table 4.1), that is the total time of execution of the processes described in this section and in the Chapter 3.

For all the scenarios, the robot moved in (closer to the table) and then moved away, in a straight line. The following parameters were used:

- Pre-processing: *leaf_size* = 0.006 meters (6mm); *minHeight* = 0.5 meters (50cm) and *maxHeight* = 1 meters;
- Object extraction: *distance_threshold* = 0.01 meters (1cm).

The purpose of these experimental tests was to measure the processing time (results presented in Table 4.1) of the projection processes described in this chapter.

Scenario	**Original cloud projection time**	**Clusters projection time**	**Total time**
1 object	1.571 ms	0.089 ms	39.071 ms
2 objects	1.529 ms	0.091 ms	42.911 ms
3 objects	1.536 ms	0.118 ms	43.726 ms
4 objects	1.639 ms	0.143 ms	46.223 ms
5 objects	1.593 ms	0.162 ms	46.348 ms

Table 4.1: Experimental results: processing time of the projection process; 150 frames were analysed.

In the Table 4.1, it is possible to see that the process that takes longer to execute is the projection of the entire original point cloud to the 2D space. Its processing time is independent of the number of objects/clusters present in the scene. As expected, the processing time of the cluster projection process depends of the number of clusters and it is higher with the increase of the number of objects above the table (this time includes the time used for the calculation of the ROI parameters and its application for each cluster).

Figure 4.4: Projection process: example of images resulting from the projection step during the tests.

CHAPTER 5

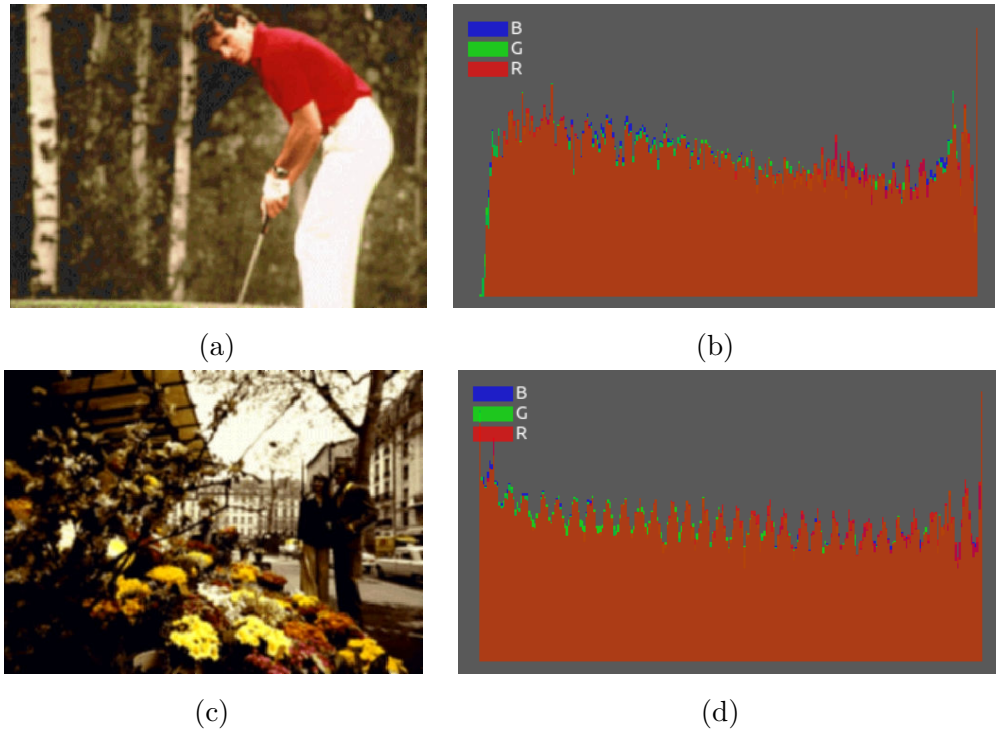
Object Recognition

This chapter describes the object recognition process implemented in the developed vision system, using visual descriptors. This process takes as input the digital images of the objects found by the processes described in the Chapter 3 and Chapter 4 and have to be capable of recognizing known objects.

5.1 visual descriptors

The use of visual descriptors is one of the techniques that can be explored for object recognition in images. These descriptors encode information of the object image and act as a signature or "fingerprint" of it, which can be used to differentiate one object from another. Ideally, the descriptors would be invariant under image transformations and illumination variations.

This work studies two approaches for object recognition for a service robot based on color histograms and feature detection algorithms. The first approach was the use of color histograms. However, a color histogram provides only a weak characterization of an image for the purpose of object recognition, because images with similar histograms can have dramatically different appearances. For example, the images shown in Figure 5.1 have similar color histograms. Later, to improve the recognition ratio of the developed system, two others algorithms were explored for feature detection and description (SIFT and SURF).



(a) (b)

(c) (d)

Figure 5.1: Two images with similar color histograms.

5.1.1 COLOR HISTOGRAMS

In image processing, a color histogram (see as example Figure 5.2) is a representation of the number of pixels of each color in an image. For digital images, a color histogram is a simple histogram that shows the color level for each individual color channel.

The color histogram can be built for any kind of color space, although the term is more often used for three-dimensional spaces like RGB or HSV. For monochromatic images, the term intensity histogram may be used instead.

If the set of possible color values is sufficiently small, each of those colors may be placed on a range by itself; then the histogram is merely the count of pixels that have each possible color. Most often, the space is divided into an appropriate number of bins, often arranged as a regular grid, each containing many similar color values.

The histogram provides a compact summarization of the distribution of data in an image.

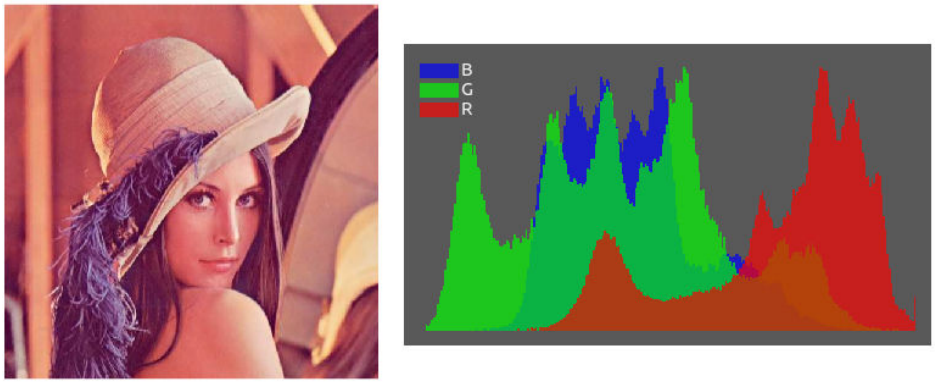


Figure 5.2: RGB histograms of the example image.

5.1.2 SIFT - SCALE INVARIANT FEATURE TRANSFORM DESCRIPTOR

Scale Invariant Feature Transform (SIFT) [5] is a popular image matching algorithm in computer vision, developed by David G. Lowe at University of British Columbia, in 2004.

This algorithm is used to detect and describe local features in an image and it is applicable in areas as object recognition, video tracking, image stitching and 3D modelling.

The SIFT descriptor is invariant to rotations, translations and scaling transformations in the image domain. It is also robust to changes in illumination, noise and perspective transformations.

For any object in an image, interesting points on the object can be extracted to provide a feature description. In the SIFT algorithm, this points are extracted (termed keypoints) from a reference image and stored in an appropriate structure (see as example Figure 5.3). Once found all keypoints, the algorithm computes a descriptor vector for each keypoint. The resulting descriptors can be compared with the others descriptors obtained from different images, in order to found matching pairs.

The SIFT method operate in a stack of gray-scale images with increasing blur, obtained by the convolution of the initial image with a variable-scale Gaussian. A differential operator is applied in the scale-space, and candidate keypoints are obtained by extraction the extrema of this differential. Position and scale of detected points are refined, and possible unstable detections are discarded. The SIFT descriptor is built based on local image gradient magnitude and orientation at each sample point in a region around the keypoint location. The descriptor encodes the spatial gradient distribution over a keypoint neighborhood using a 128-dimensional vector.

Figure 5.3: Example of keypoints detection using SIFT algorithm.

5.1.3 SURF - SPEEDED UP ROBUST FEATURE DESCRIPTOR

Speeded Up Robust Feature (SURF) [6] is a local feature detector and descriptor, presented by Herbert Bay in 2006, commonly used in tasks such as object recognition, registration, classification and 3D reconstruction. It is partly inspired by the SIFT method and it is also invariant to scale, rotation and translation.

The SURF algorithm uses integral images in the convolution process, useful to speed up this method. The box-space is constructed by using box filters approximation, by convolving of the initial images with box filters at several different discrete size. To select interest point candidates, the local maxima of a Hessian matrix is computed and a quadratic interpolation is used to refine the location of candidate keypoints (see as example Figure 5.4). Contrast sign of the interest point are stored to construct the keypoint descriptor. Finally, the dominant orientation of each keypoint is estimate and vector descriptor is computed.

Figure 5.4: Example of keypoints detection using SURF algorithm.

5.2 MATCHING

5.2.1 COLOR HISTOGRAM COMPARISON

Color histograms are frequently used to compare images. They are popular because they are trivial to compute, and tend to be robust against small changes in camera viewpoint [24].

The OpenCV library provides a method [25], first introduced by Swain and Ballard [26] and further generalized by Schiele and Crowley [27], that provides the ability to compare two histograms in terms of some specific criteria for similarity.

The four available options are the following:

- **Correlation**

$$d_{correl}(H_1, H_2) = \frac{\sum_i {H_1}'(i) \cdot {H_2}'(i)}{\sqrt{\sum_i {H_1}'^2(i) \cdot {H_2}'^2(i)}} \tag{5.1}$$

where $H'_k(i) = H_k(i) - (1/N)(\sum_j H_k(j))$ and N equals the number of bins in the histogram.

For correlation, a high score represents a better match than a low score. A perfect match is 1 and a maximal mismatch is –1; a value of 0 indicates no correlation (random association).

- **Chi-square**

$$d_{chi-square}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \tag{5.2}$$

For chi-square, a low score represents a better match than a high score. A perfect match is 0 and a total mismatch is unbounded (depending on the size of the histogram).

- **Intersection**

$$d_{intersection}(H_1, H_2) = \sum_i min(H_1(i), H_2(i)) \tag{5.3}$$

For histogram intersection, high scores indicate good matches and low scores indicate bad matches. If both histograms are normalized to 1, then a perfect match is 1 and a total mismatch is 0.

- **Bhattacharyya distance**

$$d_{bhattacharyya}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}}} \tag{5.4}$$

For Bhattacharyya matching [28], low scores indicate good matches and high scores indicate bad matches. A perfect match is 0 and a total mismatch is a 1.

5.2.2 DESCRIPTORS MATCHING

Feature matching is a fundamental aspect of many problems in computer vision, including object or scene recognition. The objective is to select the best candidate match for each keypoint. It is important to select an algorithm to perform the matching as quickly as possible and in an efficient way.

BRUTE FORCE MATCHER

Brute-Force matcher is a simple algorithm: it takes the descriptor of one feature in the first set and matched it with all other features in a second set using some distance calculation (for example, euclidean distance), returning the closest one.

FLANN - FAST LIBRARY FOR APPROXIMATE NEAREST NEIGHBOR

FLANN [29] is a algorithm for performing fast approximate nearest neighbor searches in large datasets and high dimensional spaces. It contains a collection of algorithms optimized for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

5.3 PROPOSED APPROACH

Up to this point, the potential objects present in the captured scene were detected and projected to the 2D space. As defined in Section 1.1, the developed vision system has to be capable of recognizing these objects. In order to satisfy this requirement, a set of images has to be passed as input to the system (this set of images is referred in this document as an image database). The database contains the list of object images that the system has to be capable of recognize, i.e., the objects that the system know.

The Figure 5.5 illustrates the workflow performed by the system to attempt the recognition of the found objects.

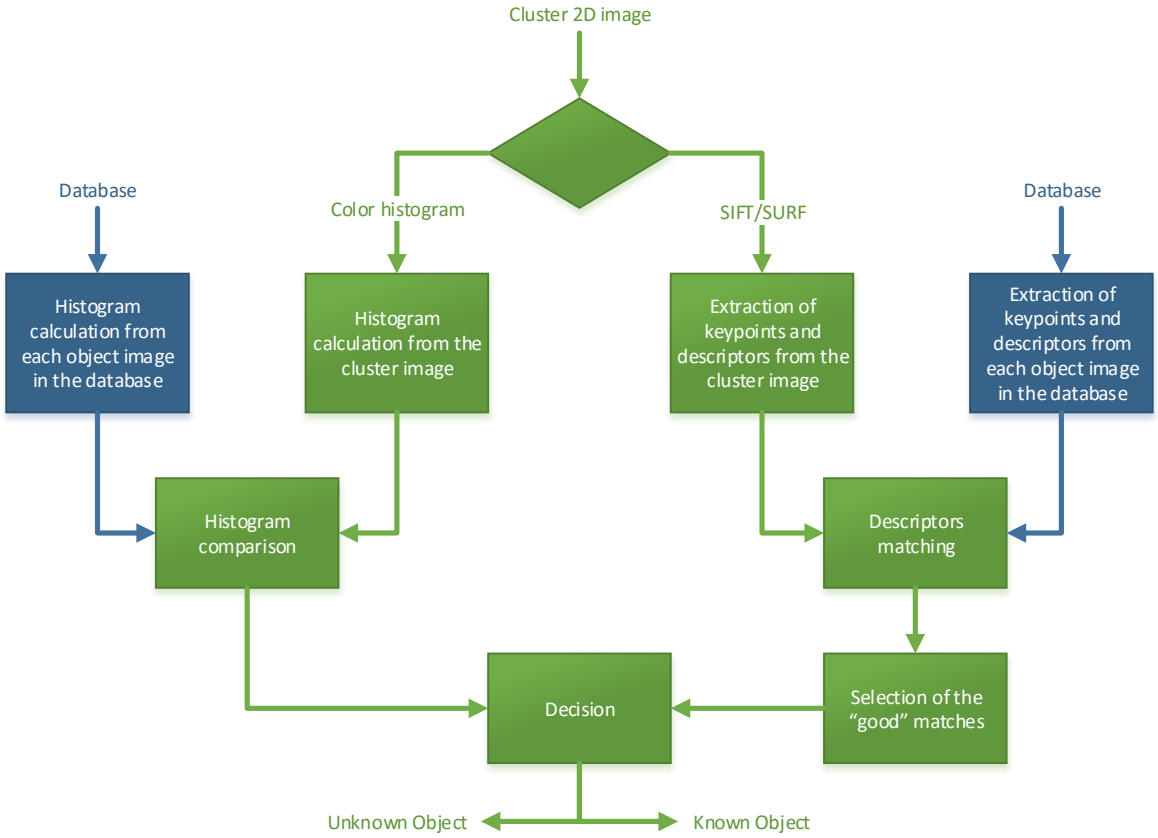


Figure 5.5: Object Recognition: structure for the proposed approach. In the image, the two "Database" items refer to the same database.

The developed vision system is provided with the three visual description algorithms described in the above sections of this chapter (color histograms, SIFT and SURF). Only one descriptor algorithm is selected at the beginning of the system execution.

In Figure 5.5, components with blue color are related to processes that run only once and at the beginning of the system execution. All the other components are executed for each found object.

Some of the processes/components have already been explained in the previous sections. For SIFT and SURF implementations, the descriptors matching component can produce a large number of matches and in some cases. For this reason, a selection process of "good" matches had to be implemented, where a reference threshold is applied and only matches with a distance less than that thresold are considered a "good" match.

In the end, for any of the cases, a process is needed to decide if a given object is known by the system (if it is present in the database).

5.4 RESULTS

5.4.1 STUDY OF THE METHODS FOR COLOR HISTOGRAM COMPARISON

The results presented in the Table 5.1 were based on the application of methods for color histogram comparison described in the subsection 5.2.1.

In this experimental test, a database with three images were used (see Figure 5.6) and four scenarios were tested, each one with only one object placed on the top of a table.

In Table 5.1:

- Method 1: it refers to Correlation;
- Method 2: it refers to Chi-square;
- Method 3: it refers to Intersection;
- Method 4: it refers to Bhattacharyya distance;
- Scenario 1: Milk package placed on the top of the table;
- Scenario 2: Cereal box placed on the top of the table;
- Scenario 3: Book placed on the top of the table;

	Method 1			**Method 2**			**Method 3**			**Method 4**		
	Milk	**Cereals**	**Book**	**Milk**	**Cereals**	**Book**	**Milk**	**Cereals**	**Book**	**Milk**	**Cereals**	**Book**
Scenario 1 (Milk)	0.0059	0.0003	-0.0009	142430	71699	50965	1753	2026	2837	0.759	0.757	0.765
Scenario 2 (Cereals)	-0.0067	0.9492	0.0288	125112	16161	1132710	2072	12961	7480	0.837	0.314	0.582
Scenario 3 (Book)	-0.0038	0.086	0.6521	113634	75613	25573	1808	6897	13258	0.858	0.633	0.332

Table 5.1: Color histogram comparison - study of methods.

In Table 5.1, the gray cells represents the best correspondence between each scenario and each method. The method 1 is the one that presents the best results for the three scenarios, although the result for scenario 1 using the method 1 is almost inconclusive (approximately zero).

(a) (b) (c)

Figure 5.6: Images database for the study of the methods for color histogram comparison: (a) Book; (b) Cereals box; (c) Milk package.

5.4.2 STUDY OF THE DESCRIPTORS MATCHING ALGORITHMS

To validate the implementation of the descriptors matching algorithms, several tests had been performed. The matching algorithm used is the FLANN and it was calculed the number of matches and "good" matches between the objects in each test scenario and the database (the datababe used was the one that is illustrated in Figure 5.6).

The test scenario consists of:

- Scenario 1: Milk package placed on the top of a table;
- Scenario 2: Cereal box placed on the top of a table;
- Scenario 3: Book placed on the top of a table;

For all the cases, a set of 150 frames were analysed, with the robot describing a movement like the one illustrated in the Figure 5.7.

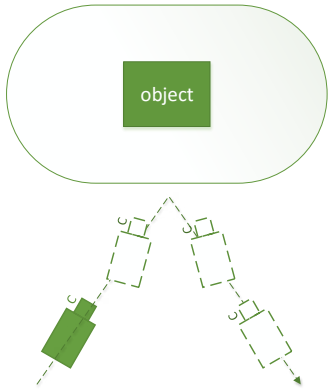


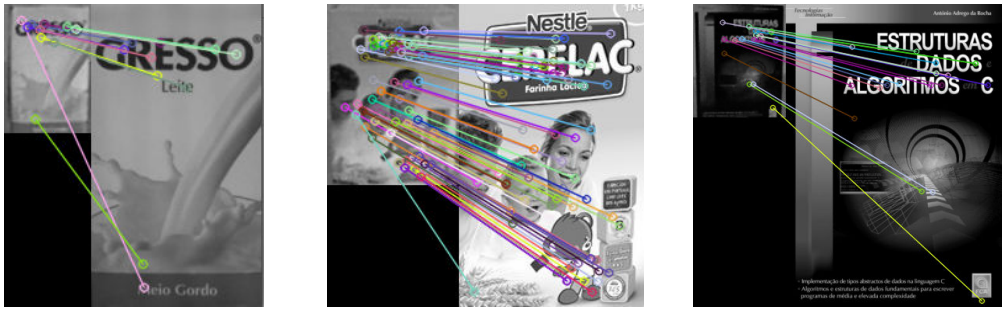
Figure 5.7: The green rounded rectangle represents a table with an object. The dashed line represent the movement of robot. During all process, the Kinect was directed to the table. IThis movement allow to study the influence of variation in scale and rotation.

The selection of "good" matches in the developed system, was inspired in [5]. The system rejects all the matches in which the distance ratio is greater than 0.8, which (according to the author) eliminates 90% of the false matches while discarding less than 5% of the correct matches.

The Figure 5.8 and Figure 5.10 are visual examples of the execution of the matching algorithms. In all these illustrated cases, the larger images correspond to the image stored in the database and the smaller, correspond to the image of the object detect by the developed vision system.

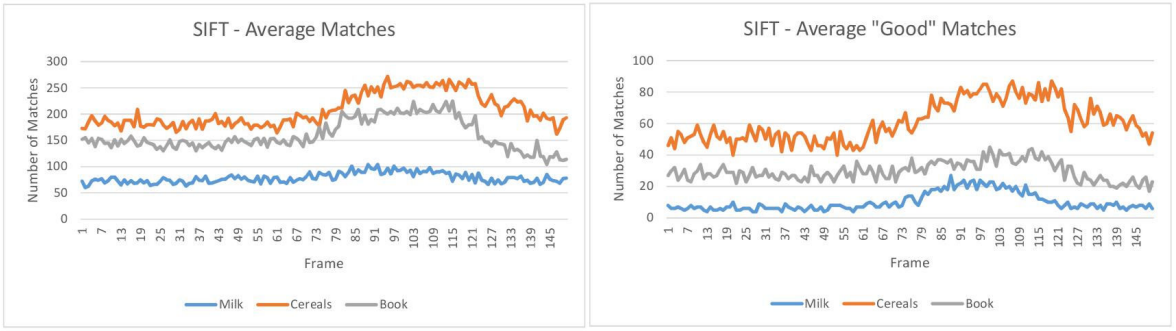
The Figure 5.9 and Figure 5.11 represents the number of matches and "good" matches for each scenario in each frame, for SIFT and SURF implementations respectively.

SIFT



(a) (b) (c)

Figure 5.8: SIFT descriptors matching: Colored lines represent the "good" matches; (a) scenario 1; (b) scenario 2; (c) scenario 3.

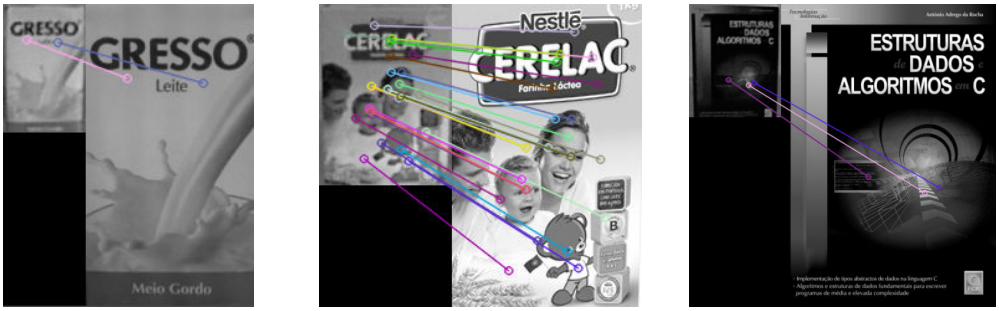


(a) (b)

Figure 5.9: SIFT descriptors matching: (a) Number of matches; (b) Number of "good matches".

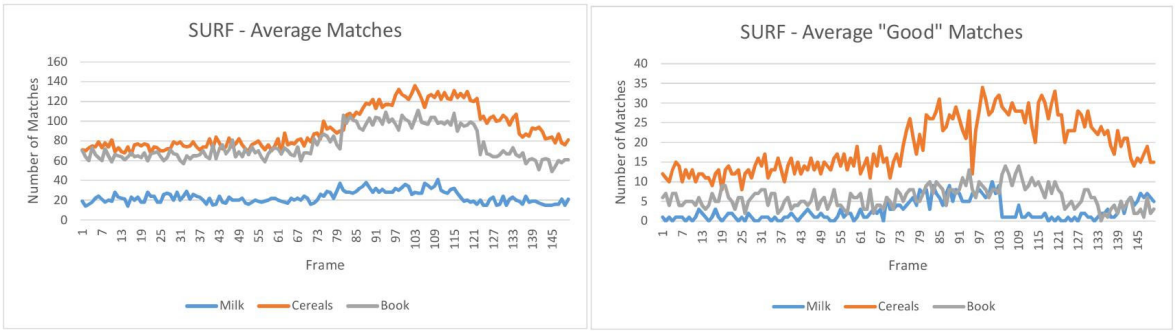
In Figure 5.8 and Figure 5.9 it is possible to observe that the scenario 1 is the one that have the smallest number of matches. It can be notted that only a small number of matches can be extracted for the milk package and this is due to the lack of texture and details of this object.

SURF



(a) (b) (c)

Figure 5.10: SURF descriptors matching: Colored lines represent the "good" matches; (a) scenario 1; (b) scenario 2; (c) scenario 3.



(a) (b)

Figure 5.11: SURF descriptors matching: (a) Number of matches; (b) Number of "good" matches.

In Figure 5.10 and Figure 5.11 it is possible to observe that the scenario 1 is the one that have the smallest number of matches. Comparatively with the SIFT, the SURF algorithm presented a smaller number of matches and "good" matches, most visible in the case of the milk package and of the book as well.

For all the scenarios in the Figure 5.9 and Figure 5.11, the increasing number of matches and "good matches" observed in the graphics can be justified with the fact that the robot moved in the direction of the table and captured the objects with higher resolution.

5.4.3 STUDY OF THE DECISION PROCESS

Given the matching results and for each of the implemented algorithms (histogram, SIFT and SURF), a decision has to be taken about if a specific object is or not known by the system.

In the histogram comparision, the chosen method was the method 1 (correlation). All the found objects have to be compared with the objects present in the database and it was defined that results greater than 0.2 have to be considered as a recognition. If more than one correlation result is present, the result with higher value is chosen.

In the SIFT implementation, it was defined, by observation of the results presented in the Figure 5.9, that a minimum number of "good" matches have to be present so that the object is considered a recognition. The value defined was a minimum of 5 "good" matches.

For the SURF implementation, the some process was used and a minimum value of 2 "good" matches is defined.

Also for these two last algorithms, if more than one result passed the defined value, the result with higher value is the chosen.

5.4.4 EFFICIENCY

To evaluate the processing time of the object recognition algorithm in the developed vision system, some experiments were conducted with the CAMBADA@Home robot. The scenario chosen was the same that is described in the Section 3.6 and a database with three images were used (see Figure 5.6).

Some tests were performed and they only differ in the number of objects lying on the table. The results present in this section only take in account the work described in this section, except the overall processing time, that is the total time of execution of all the developed system.

For all the scenarios, the robot moved in (closer to the table) and then moved away, in a straight line. The following parameters were used:

- Pre-processing: *leaf_size* = 0.006 meters (6mm); *minHeight* = 0.5 meters (50cm) and *maxHeight* = 1 meters;
- Object extraction: *distance_threshold* = 0.01 meters (1cm).

The purpose of these experimental tests was to measure the efficiency of object recognition process described in this chapter in all the implemented algorithms. The Table 5.2 presents the results for the histogram comparison, the Table 5.3 represents the results for the SIFT implementation and the Table 5.4 presents the results for the SURF implementation.

Scenario	Classification time	Total time
1 object	1.428 ms	41.269 ms
2 objects	2.887 ms	45.013 ms
3 objects	5.259 ms	48.893 ms
4 objects	7.321 ms	50.061 ms
5 objects	9.913 ms	55.925 ms

Table 5.2: Experimental results: efficiency of the histogram comparison process; 150 frames were analysed.

Scenario	Classification time	Total time
1 object	24.428 ms	64.437 ms
2 objects	39.782 ms	85.062 ms
3 objects	57.207 ms	102.119 ms
4 objects	74.686 ms	119.283 ms
5 objects	88.623 ms	132.823 ms

Table 5.3: Experimental results: efficiency of the SIFT algorithm implementation; 150 frames were analysed.

Scenario	Classification time	Total time
1 object	13.698 ms	52.142 ms
2 objects	20.678 ms	63.492 ms
3 objects	30.574 ms	74.262 ms
4 objects	36.512 ms	79.891 ms
5 objects	39.137 ms	83.138 ms

Table 5.4: Experimental results: efficiency of the SURF algorithm implementation; 150 frames were analysed.

As can be seen in the results presented in the table 5.2, 5.3 and 5.4 the processing time of each classification algorithm is dependent of the number of the clusters and it is higher as the number of objects above the table.

The most efficient algorithm, i.e., the one with the smaller processing time is the histogram comparison algorithm and the less efficient is the SIFT implementation.

The study of the recognition ratio of each classification algorithm will be described in the Chapter 6.

CHAPTER 6

Experimental Results

This chapter describes a case test to evaluate the developed vision system. Other test cases were used and described along this document, but they were intended to validate only the work done in each of these chapters. This one, was designed to evaluate the effectiveness of the entire developed vision system.

6.1 Test Scenario

Some challenges in the RoboCup@Home league involve the detection and recognition of objects. These objects resemble items usually found in household environments like, for instance, soda cans, coffee mugs or books. Robots should be able to navigate between all divisions of the simulated household environment - Figure 6.1.

(a) (b)

Figure 6.1: Typical house environment from RoboCup@Home 2015 competition: (a) Map; (b) Environment.

To evaluate the efficiency and the recognition ratio of the developed vision system, some experiments were conducted with the CAMBADA@Home robot.

Some objects were selected and several acquisitions (bag files [1]) were acquired using the Kinect installed on the robot. The objects database used in these experiments are represented in Figure 6.2. The scenario chosen was a table where the objects were placed and the robot performing the movement illustrated in the Figure 5.7, varying the scale and rotation of the camera relatively to the position of objects.

(a) (b) (c)

Figure 6.2: Images database for the experimental tests: (a) Book; (b) Cereals box; (c) Milk package.

6.2 RESULTS

To evaluate the efficiency and recognition ratio of the developed vision system, some experiments were conducted with the CAMBADA@Home robot. The scenario chosen for the experiments is a common table and the objects are items usually found in household environments.

Some tests were performed and they only differ in the number of objects lying on the table.

The results present in this section were obtained using the following parameters:

- Pre-processing: *leaf_size* = 0.006 meters (6mm); *minHeight* = 0.5 meters (50cm) and *maxHeight* = 1 meters;
- Object extraction: *distance_threshold* = 0.01 meters (1cm).

[1] http://wiki.ros.org/Bags

Several scenarios were recorded:

- Scenario 1: All objects in the scene;
- Scenario 2: All objects except the book;
- Scenario 3: All objects except the cereals box;
- Scenario 4: All objects except the milk package;
- Scenario 5: All objects except the soda can;

It is important to note that the soda can object don't have a correspondence in the database. This mean that this object should be unknown to the vision system and it don't should be recognized.

The histogram algorithm was not tested in this chapter because the results obtained were not satisfactory and were fairly inconclusive.

The Figure 6.3 illustrates an example of the scenario 1 used for the experimental tests.

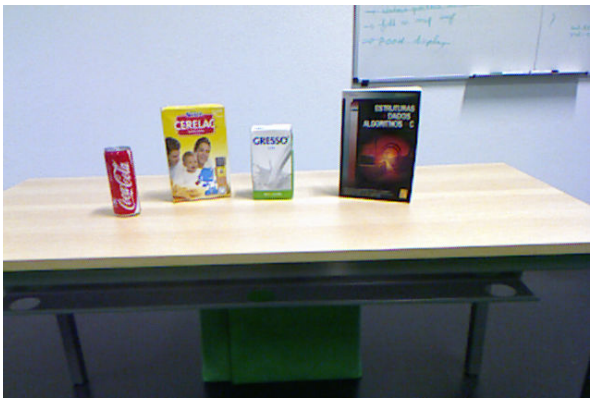


Figure 6.3: Example of the scenario used in the experiments: scenario 1.

SIFT

In this subsection, the results presented were obtained with execution of the vision system with the SIFT implementation.

	Voxel time	**Pre-processing time**	**Segmentation time**	**Clustering time**	**Projection time**	**Classification time**	**Total time**
Scenario 1	31.691 ms	3.596 ms	3.211 ms	19.502 ms	2.203 ms	149.455 ms	213.745 ms
Scenario 2	31.871 ms	3.714 ms	2.751 ms	15.167 ms	2.133 ms	109.979 ms	169.351 ms
Scenario 3	32.080 ms	3.587 ms	2.911 ms	16.443 ms	2.306 ms	109.418 ms	170.158 ms
Scenario 4	31.998 ms	3.949 ms	3.032 ms	19.011 ms	2.292 ms	122.926 ms	187.077 ms
Scenario 5	32.389 ms	3.513 ms	3.289 ms	19.888 ms	2.282 ms	136.133 ms	201.818 ms

Table 6.1: Experimental results: efficiency of the developed system using the SIFT algorithm; 250 frames were analysed; the processing time of the plane alignment was not considered.

Scenario	Book	Cereals	Milk
Scenario 1	91.2 %	99.2 %	75.6 %
Scenario 2	0%	100 %	95.2 %
Scenario 3	86.8 %	3.6 %	92.8 %
Scenario 4	88.4 %	98.8 %	1.2 %
Scenario 5	87.2 %	96.8 %	87.2 %

Table 6.2: Experimental results: Recognition rate regarding the SIFT algorithm, gray cells corresponds to false positives. 250 frames were analyzed.

SURF

In this subsection, the results presented were obtained with execution of the vision system with the SURF implementation.

	Voxel time	Pre-processing time	Segmentation time	Clustering time	Projection time	Classification time	Total time
Scenario 1	31.649 ms	3.312 ms	3.131 ms	18.327 ms	2.267 ms	101.705 ms	164.268 ms
Scenario 2	30.123 ms	3.278 ms	2.727 ms	14.261 ms	2.136 ms	72.559 ms	128.259 ms
Scenario 3	31.002 ms	3.417 ms	2.719 ms	15.946 ms	2.191 ms	72.219 ms	130.576 ms
Scenario 4	31.433 ms	3.646 ms	3.015 ms	18.579 ms	2.138 ms	80.303 ms	142.492 ms
Scenario 5	32.013 ms	3.630 ms	3.595 ms	19.218 ms	2.171 ms	91.219 ms	155.891 ms

Table 6.3: Experimental results: efficiency of the developed system using the SURF algorithm; 250 frames were analysed; the processing time of the plane alignment was not considered.

Scenario	Book	Cereals	Milk
Scenario 1	90 %	97.6 %	72 %
Scenario 2	0%	99.6 %	83.6 %
Scenario 3	86 %	0%	74.4 %
Scenario 4	78 %	99.2 %	2%
Scenario 5	84.4 %	99.2 %	68.4 %

Table 6.4: Experimental results: Recognition rate regarding the SURF algorithm, gray cells corresponds to false positives. 250 frames were analyzed.

During the execution of the tests, it was observed that the best results are obtained when the robot is located in front of the objects and close to them. This is due to the fact that the image resolution of these objects decrease with distance.

As can be observed in Table 6.1 and Table 6.3, the SURF algorithm presents a lower processing time, when compared to the SIFT algorithm. Although, in Table 6.2 and Table 6.4, the presented results are slightly more accurate in the SIFT algorithm than in the SURF implementation.

The compromise between the processing time and the recognition rate confers that the SURF implementation should be the best choise.

CHAPTER 7

Conclusions and future work

7.1 conclusions

The objective of the work presented in this thesis was the development of a computer vision system for a service robot, wherein the only focus of the system was the detection and recognition of objects.

A solution was achieved and it presents two main stages of processing: object detection and object classification for recognition purposes.

In the first stages of the execution, the point cloud of the captured scene is processed in order to extract objects that lie on horizontal planes, such as on the ground or on a table. Initially, the point cloud is preprocessed to filter some existing noise and to reduce its size. This initial filtering allows the reduction of processing time of the following stages. Finalized this stage, the most dominant plane is segmented and the points above the segmented plane are extracted. Using the resulting points, a clustering process is performed in order to isolate individual clusters. The retrieved clusters represent individual objects detected by the developed vision system.

For object recognition, two approaches were studied. Both approaches rely on the existence of a previously built database of images containing different objects of interest. The first method is based on the match between color histograms, while the second one is based on image descriptors (SIFT and SURF). Both methods work with digital images. So, the resulting clusters from the object detection stage have to be projected to 2D images. Given the projected images, the vision system is capable of using one of the implemented approaches to compare the detected objects with the images present in the database. It was observed that the color histogram approach was the method that presents lower processing time, but being a limited algorithm, because images with similar histograms can have dramatically different appearances. Comparing the image descriptors algorithms, the SURF algorithm presents

a lower processing time, when compared to the SIFT algorithm. However, the results are slightly more accurate in the SIFT algorithm than in the SURF implementation, which means that there is a compromise that a user of this system should do in order to decide whether time processing or object recognition rate is more important in a given application.

In conclusion, the proposed vision system presents characteristics that deem it capable of performing detection and recognition of household objects in domestic environments. By the end of this work, the developed vision system was included in the CAMBADA@Home robot satisfying all the initial objectives.

7.2 FUTURE WORK

Given that the CAMBADA@Home platform did not have a vision system capable for detecting objects, this project acts as a basis for future applications in this field. Therefore, it is important to point some modules that would be improved and some algorithms that can be implemented:

- The study and improvement of this work with more objects and under different perspectives should be one of the focus on the future;
- The ability to learn new objects on-line should be developed, extracting the visual descriptors of the objects and storing them in the database;
- The proposed vision system has methods to segment horizontal surfaces, where most of the objects of interest are expected be found. It works well for surfaces like tables and the ground floor, but it should be improved with the capability of segment others interesting surfaces, like chairs and shelves;
- Depth information should be used in the object recognition module so that new feature extraction algorithms can be explored and implemented, i.e., using 3D data instead of 2D images., and including the combination and integration of 3D models.

Bibliography

[1] *Robot Operating System Official Website*. [Online]. Available: `http://www.ros.org/` (visited on 07/20/2015).

[2] P. Aleixo, "Object detection and recognition for robotic applications", Master's thesis, University of Aveiro, 2014.

[3] L. Ferreira, "CAMBADA@Home: Detection and Tracking of Humans", Master's thesis, University of Aveiro, 2013.

[4] P. Loncomilla and J. Ruiz-del-solar, "Object recognition for manipulation tasks in real domestic settings: A comparative study",

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, 2006, pp. 404–417.

[7] X. Chen, D. Lu, K. Chen, Y. Chen, and N. Wang, "KeJia : The Intelligent Service Robot for RoboCup @ Home 2014", 2014.

[8] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation", *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, pp. 47–54, 2009, ISSN: 08966273. DOI: `10.1109/ICCVW.2009.5457718`.

[9] R. B. Rusu and S. Cousins, "3D is here: point cloud library", *IEEE International Conference on Robotics and Automation*, 2011. DOI: `10.1109/ICRA.2011.5980567`. [Online]. Available: `http://pointclouds.org/`.

[10] R. Dwiputra, M. Füller, F. Hegger, S. Schneider, I. Awaad, J. M. S. Loza, A. Y. Ozhigov, S. Biswas, N. V. Deshpande, A. Hagg, I. Ivanovska, P. G. Ploeger, and G. K. Kraetzschmar, "The b-it-bots RoboCup@Home 2014 Team Description Paper", *RoboCup, João Pessoa, Brazil*, no. Figure 3, pp. 1–6, 2014.

[11] C. a. Mueller, N. Hochgeschwender, and P. G. Ploeger, "Surface Reconstruction with Growing Neural Gas", *IROS'12 Workshop on Active Semantic Perception (ASP'12)*, pp. 1–5, 2012.

[12] J. J. M. Lunenburg, S. A. M. Coenen, and T. T. J. Derksen, "Tech United Eindhoven @ Home Team Description 2014", 2014.

[13] T. D. Jager, "Robust object detection for service robotics", Master's thesis, Utrecht University, 2013.

[14] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System", *Icra*, 2009, ISSN: 0165022X. DOI:

http://www.willowgarage.com/papers/ros-open-source-robot-operating-system. [Online]. Available: http://pub1.willowgarage.com/~konolige/cs225B/docs/quigley-icra2009-ros.pdf.

[15] M. Schwarz, J. Stückler, D. Droeschel, K. Gräve, D. Holz, M. Schreiber, and S. Behnke, "NimbRo@Home 2014 Team Description", [Online]. Available: http://www.nimbro.net/@Home.

[16] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking", *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 137–147, 2014, ISSN: 10473203. DOI: 10.1016/j.jvcir.2013.02.008.

[17] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from RGB-D video", *Journal of Real-Time Image Processing*, no. 2012, pp. 1–11, 2013, ISSN: 18618200. DOI: 10.1007/s11554-013-0379-5.

[18] J. Cunha, A. Neves, J. Azevedo, and B. Cunha, "A mobile robotic platform for elderly care", *1st International Living Usability Lab Workshop on AAL Latest Solutions, Trends and Applications (In conjuction with BIOSTEC), AAL'11*, pp. 36–45, 2011. [Online]. Available: http://www.ieeta.pt/atri/CAMBADA/athome/docs/Cunha-2011a.pdf$%5Cbackslash$nhttp://www.biostec.org/BIOSTEC2011/AAL.asp$%5Cbackslash$nhttp://wiki.ieeta.pt/wiki/index.php/Cunha-2011c.

[19] L. Seabra Lopes, "Carl, a Learning Robot, serving Food at the AAAI Reception", *Proc. of the AAAI Mobile Robot Competition and Exhibition Workshop*, pp. 1–7, 2001.

[20] J. Cunha, J. L. Azevedo, M. B. Cunha, L. Ferreira, P. Fonseca, N. Lau, C. Martins, A. J. R Neves, E. Pedrosa, A. Pereira, L. Santos, and A. J. S Teixeira, "CAMBADA@Home'2013: Team Description Paper",

[21] G. Bradski, "The opencv library", *Dr. Dobb's Journal of Software Tools*, 2000.

[22] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applicatlons to Image Analysis and Automated Cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981, ISSN: 0001-0782. DOI: 10.1145/358669.358692. [Online]. Available: http://dx.doi.org/10.1145/358669.358692.

[23] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments", *KI - Künstliche Intelligenz*, pp. 1–4, 2010, ISSN: 09331875. DOI: 10.1007/s13218-010-0059-6.

[24] G. Pass and R. Zabih, "Histogram refinement for content-based image retrieval", *IEEE Workshop on Applications of Computer Vision*, pp. 96–102, 1996. DOI: 10.1109/ACV.1996.572008.

[25] A. Kaehler and B. Gary, *Learning OpenCV*. 2013, p. 575, ISBN: 978-1-4493-1465-1. DOI: 10.1109/MRA.2009.933612.

[26] M. J. Swain and D. H. Ballard, "Color Indexing", *International Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[27] B. Schiele and J. L. Crowley, "Object recognition using multidimensional receptive field histograms", *Lecture Notes in Computer Science*, vol. 1064, no. section 6, pp. 610–619, 1996.

[28] A. Bhattacharyya, "On A Measure of Divergence Between Two Statistical Populations Defined by their Probability Distributions", *Bulletin of Cal. Math. Soc.*, vol. 35, no. 1, pp. 99–109, 1943.

[29] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration", *In VISAPP International Conference on Computer Vision Theory and Applications*, pp. 331–340, 2009.