Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Facultat d'Informàtica de Barcelona (FIB)

Vision and learning algorithms for service robots :
the task of welcoming visitors

Bachelor Degree in Informatics Engineering

Specialization: Computing

Laia Freixas

Thesis supervisor
Dr. Guillem Alenyà
IRI (CSIC-UPC)

Tutor
Dr. Javier Vázquez
LSI

Monday 2nd July, 2018

Abstract

Service robotics is a growing field in technology. We are starting to wonder what are the benefits of having robots that carry out tasks that are usually done by humans. Amongst these tasks, we specifically focus on identifying and welcoming guests who visit our home. In the context of the European Robotics League Service robot competition, we developed software which allowed a TIAGo robot to identify and interact with four different visitors, by recognizing their attire and face. This was done by studying the current algorithms used for face detection and recognition, as well as using different vision techniques such as color segmentation. Furthermore, a state machine was developed for the robot to do different tasks and interact differently depending on which person was visiting. The project was tested in two local rounds of the competition, and the results were satisfactory, by obtaining a high accuracy in person detection and recognition, as well as a robust state machine which allowed to win the prize for this task in the competition.

Resumen

El campo de los robots de servicio está creciendo en el sector tecnológico, lo que causa que empecemos a preguntarnos sobre los beneficios de que haya robots que desarrollen tareas y actividades de las que normalmente se encargan los humanos. Entre estas actividades, nos centramos específicamente en identificar y dar la bienvenida a las personas que visitan nuestra casa. En el contexto de la competición European Robotics League para robots de servicio, hemos desarrollado un *software* que permite al robot TIAGo identificar e interactuar con cuatro visitantes distintos, usando información de su ropa y su cara para reconocerlos. Esto fue desarrollado a partir del estudio de los algoritmos que se utilizan para detección y reconocimiento de caras, así como usando algunas técnicas de visión, como segmentación de color. Además, se desarrolló una máquina de estados para que el robot pudiera hacer diferentes acciones e interactuar de forma distinta dependiendo de qué persona estuviera visitando. El proyecto fue evaluado en dos rondas locales de la competición, y los resultados fueron satisfactorios, ya que se obtuvo una alta precisión en detección y reconocimiento de caras, así como una máquina de estados robusta que permitió ganar el premio para esta parte de la competición.

Resum

El camp dels robots de servei està tenint un gran creixement en la tecnologia, i estan començant a aparèixer debats sobre quins són els beneficis de tenir robots que facin les tasques que fins ara han fet les persones. Entre aquestes tasques, ens centrem específicament en reconèixer i rebre les persones que arriben a la nostra casa. En el context de la competició European Robotics League per a robots de servei, hem desenvolupat un *software* que permet a un robot TIAGo reconèixer i interactuar amb quatre visitants diferents, a partir de la identificació de la seva roba i la seva cara. Tot això es va aconseguir a partir de l'estudi dels mètodes actuals de detecció i reconeixement de cares, alhora que utilitzant diferents tècniques de visió per computador, com per exemple, la segmentació de color. A més, vam crear una màquina d'estats perquè el robot pogués fer diverses tasques i interaccionés de forma diferent depenent de quina de les persones volia entrar al pis. El projecte va ser avaluat en dues rondes locals de la competició, i els resultats van ser satisfactoris, ja que es va aconseguir una precisió alta en la detecció i reconeixement de cares, alhora que una màquina d'estats robusta que va permetre obtenir el premi per a aquesta part de la competició.

Contents

Chapter 1

Introduction

1.1 Project formulation and context

This project's formulation was to design and implement software on a TIAGo robot (Figure 1.1), so that it is able to identify and recognize different people. In order to do this, computer vision and machine learning algorithms were used; since they have been proven to be the best methods with which to tackle image recognition problems.

TIAGo is a robot developed by the PAL Robotics company. It gets its name from the acronym "take it and go". It is a robot which has autonomous navigation, perception, and manipulation capabilities, which makes it ideal for human collaboration tasks. TIAGo is programmed with ROS[1], and it has a modular hardware architecture which allows it to be programmed for many different tasks. It can move at a speed of up to 1 meter per second, and it has a RGBD camera and a microphone. It has an arm with 7 degrees of freedom which can pick up objects of up to 2 kg in weight. At the end of its arm it can either have a hand with five fingers or a gripper, which helps with grasping different household objects. There are three versions of TIAGo, which are iron, steel and titanium. In our case, a TIAGo steel robot was used, and the full specifications of the robot are shown in Appendix B.

The software developed in this project was used as part of a defined task in a European competition, in which a service robot helped recognize the people who rang the bell of an apartment, and then responded accordingly depending on who that person was.

The developed software needed to retrieve images from an IP camera at the front door of the flat, and process it. The output was one of the four possible people which were described in the task statement, which will be explained in further detail in the following sections. Then, the robot had to navigate the flat and interact with the visitors accordingly.

[1] Robot Operating System, `http://ros.org/`

This project covers one of the pieces of software developed with the IRI[2] team in order to compete in the ERL [3]. The team was formed by four undergraduate students and three professionals in robotics. Each of the undergraduates was in charge of one of the competition tasks, and the other team members provided technical support and guidance throughout the preparation of the competition.

The software developed in this project was tested in the ERL local rounds in Barcelona (November 2017) and Edinburgh (January 2018). This competition is organized by SPARC [4], with the collaboration of PAL Robotics; who loaned the TIAGo robot to some of the teams that participated in this competition, including the IRI team.

Figure 1.1: TIAGo robot

The stakeholders can be viewed in two ways. Firstly, in the hypothetical competition world, the beneficiary would be the apartment's inhabitant who, due to some impairments, needed help with the task of receiving visitors. Secondly, in a broader context, obtaining a good result in this task would allow the IRI team to achieve more points in the competition.

1.2 Scope and drawbacks

This project focuses on researching and understanding the current algorithms used for person recognition; and then implementing a new version which can work on a TIAGo robot and is specifically designed to solve the ERL task at hand.

The aim of this project did not include developing new techniques for image processing and recognition, it was simply to adapt existing practices into a new

[2]Institute of Robotics and Industrial Informatics, `http://www.iri.upc.edu/`
[3]European Robotics League, `https://www.eu-robotics.net/robotics_league/`
[4]The Partnership for Robotics in Europe `http://eu-robotics.net/sparc/`

software, which was specifically designed for the current task we wanted to solve, and implement it on a robot using robotic frameworks, such as Robot Operative System (ROS).

As mentioned before, the IRI team developed all the necessary elements to compete in the local round of the ERL. Since the team participated in only four tasks in the competition, each of the teammates was in charge of developing one of the tasks, plus additional modules and nodes that provided useful functionalities. The tasks' description is summarized in Section 1.4. Specifically, this project was focused on developing a library to handle person recognition for the competition, creating a ROS server in order to access its functionalities and a module so that the functions of the library were accessible (See Figure 1.2). Having this, a state machine for the Task Benchmark 2 (TBM2) could be developed, which used the person classifier module amongst others, all communicated with ROS, as seen in Figure 1.3. The code developed needed to interact with other pieces of software, such as the robot's navigation or its text-to-speech module. In order to simplify these interactions, other APIs, designed by other team members, were used. An API is a module which includes different functions.

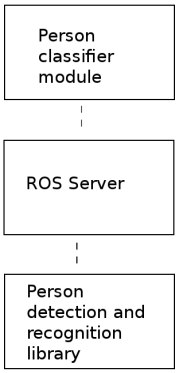


Figure 1.2: Schema of the library and ROS implementation

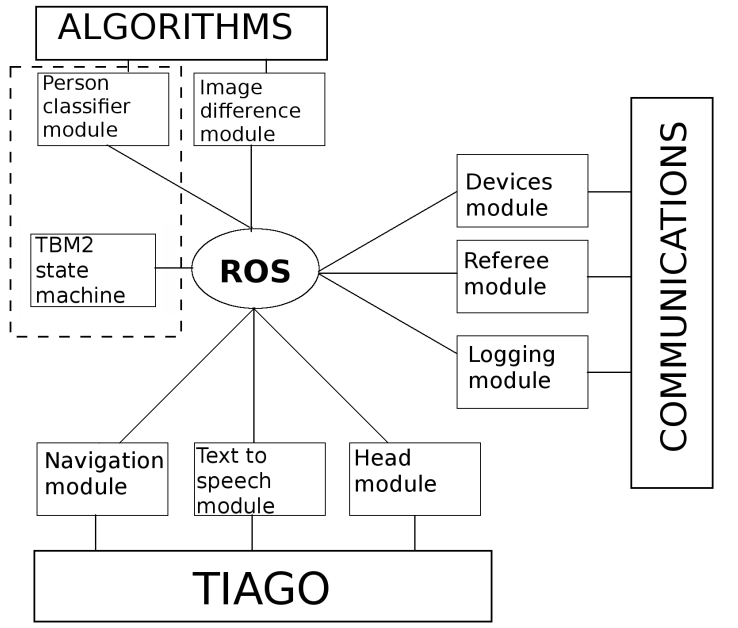


Figure 1.3: Schema of how the project fits in with the rest of modules developed. The dotted section is the one developed in this project

Before the development of the project we attempted to foresee possible difficulties that could arise during the task development.

The difficulties are the following:

Time limitation

The software for this project was developed from July to November 2017, and it was only tested from October to the end of November. This period of time was limited; and it could not be extended because the competition date was fixed. In order to overcome this, some extra hours were invested so that the project was complete before the first competition. Specific details of the original and final plan will be explained in Chapter 2

GPU requirements

Some of the machine learning algorithms require a lot of computation in order to train neural nets. The better the GPU used, the sooner the training is completed, so it was better in order to obtain results as soon as possible. Luckily, the machine used to train the neural nets had two GPUs, model NVIDIA GeForce GTX 1080 Ti.

Person face resolution

The camera resolution combined with its distance to the visitor who rang the bell could have caused some issues because the face extracted from each image had at most 150 x 150 pixels. This implied some trouble when classifying the person, as described in Section 6.2, when analyzing the results of our different approaches.

Competition tension

In every competition, a common problem is the stress or tension caused to the participants. For this project, not only was it important to have all code finished before the deadline, it was also necessary to remain calm during the competition days and to not panic if something did not go as planned.

External restrictions

The software project presented in this document is a small part of a bigger project used in a competition. This means that many decisions were not chosen independently within this project, but were set by external actors and needed to be followed. In the context of this project we modeled these external technical restrictions as project requirements.

- **Usage of ROS** The software that the TIAGo robot contains is all designed as ROS nodes, so we had to adapt our implementation to match the one included in the robot. PAL Robotics provided several previously programmed functionalities such as text to speech and basic navigation.
- **ROS Modules** The IRI development method includes using ROS modules when implementing big software projects. This allows to have a much simpler and readable code since it only includes function calls, and it does not handle the different outcomes that a ROS node introduces.
- **Competition rules** From the beginning of the project we were focused on having good results in the competition. This meant that we followed constantly the booklet provided and made our code as specific as possible for the task defined.

1.3 Motivation

There was a great interest in doing a project in the field of assistive robotics, due to the following reasons.

Mainly, it has recently been observed that the growing number of elderly population has created a greater need for people to care for them. This is primarily caused by the baby-boomer generation reaching an age over 65. Since many people are not able to afford a nursing home, new solutions have been created by using robots to take care of people in need. The duties of these

robots depend on the specific needs of the person, but most of them are common between individuals. The following duties were specified by Baltus et al. in 2000 [20], but are still significant today:

1. Cognitive prosthesis, making the robot become an assistant that reminds the person when to take their medicine, or telling them what the weather is like.

2. Safeguarding, in which the robot is used to monitor if the person is safe, and detect accidents such as falling.

3. Social interaction method, to provide the elderly person with company.

4. Transmission, in which the robot is used to communicate with a doctor via live video stream, allowing a specialist to make a diagnosis without needing to move.

5. Monitoring, in which the robot collects data and controls the activities of the person, which can be of high value in order to improve the diagnoses for treatment.

When the ERL local round in Barcelona was announced, the IRI team was excited to be able to participate in it. There were several reasons for this; firstly, it was that the thrill of the competition was very appealing to us. The second reason was the fact that we could have access to a TIAGo robot and program it and finally, the possibility of participating in a European project. Specifically, I was excited to be in charge of the person detection and recognition part of the project, because on one hand, the current research on these fields is something that interests me greatly, and on the other hand, due to my previous experience in computer vision and machine learning, I believed I could implement the algorithms effectively.

1.4 ERL and the tasks

The European Robotics League (ERL) is a set of competitions organized by euRobotics and SPARC, two European robotics companies. These competitions include ERL Service Robots, ERL Industrial Robots and ERL Emergency Robots. This project focuses exclusively on the ERL Service Robots competition, which concentrates on service robotics for home application, and was originally based on the EU-FP7 project RoCKIn@Home, which took place in the years 2014 and 2015.

The ERL consists of multiple Local Tournaments, and teams must participate in a minimum of 2 tournaments in order to be eligible for the prize. At the end of the season, a final score is calculated for each team with respect to each Task and Functionality Benchmark. Prizes for the winners are awarded during the European Robotics Forum, which this year took place in Tampere, Finland, in March 2017. [8]

The ERL service robots competition consists of five task benchmarks (TBM) and three functionality benchmarks(FBM), which are all related to possible required purposes of a real life service robot.[3] The back story is that an elderly person, named "Granny Annie", lives in an apartment. She suffers from typical problems of elderly people, such as mobility constraints. Sometimes, a doctor must come and visit her, to check on her health. She enjoys reading and socializing with friends. The robot must be able to support Granny Annie in these activities, so that her life is easier. Amongst the possible abilities, the robot is able to interact with the house's electronic devices, communicate with Annie and bring her the items she requests.

More specifically, the task benchmarks are the following :

1. TBM1 : Getting to know my home
   Navigate the flat recording what objects are changed based on an initial disposition.

2. TBM2 : Welcoming visitors
   Identify which person has rung the door bell and react appropriately for each one.

3. TBM3 : Catering for Granny Annie's Comfort
   Interact with the flat devices and bring objects to Annie based on her requests.

4. TBM4 : Visit my home
   Navigate through several way-points in the flat and do different tasks in each of them.

5. TBM5 : General Purpose Service Robots
   Understand a set of commands and do what is requested if possible.

There are also three functionality benchmarks, related to speech recognition, navigation and object recognition. Our team at IRI (CSIC-UPC) participated in tasks 1-4, and this project is focused only on TBM2.

In this task, the service robot has the goal of recognizing which person has rung the doorbell, and respond accordingly. The possible people who come in are the following :

1. The Postman, who wears a yellow uniform. The expected behavior is to greet him, ask him to leave the mail in a table in the hall and allow him to exit.

2. The Deli Man, who wears a white uniform with a logo and a cap. The expected behavior is to greet him, lead him to the kitchen, ask him to leave the breakfast on the table and allow him to exit.

3. Dr. Kimble, whose face is known but may change clothes. The expected behavior is to greet him, lead him to Annie's bedroom, wait for him to exit and follow him to the entrance door.

4. Unknown person, for whom the door must not be opened.

   Having this task in mind, the goal is to develop a set of algorithms that will solve this correctly in the TIAGo robot.

1.5 Objectives

The main objective for this project is synthesized in the following sentence.

Implement person recognition software on a service robot by using several tools and modules so that it can answer the door for different visitors

In order to achieve this goal, we have defined four measurable sub-objectives which we will use to evaluate the quality of our final software.

- **Implement a face detection algorithm, with an accuracy of at least 90%**. This objective will be reached by reading and understanding the current literature on the subject of face detection and choosing the option that provides the best results, by using different computer vision techniques.

- **Implement a person recognition algorithm, with an accuracy of at least 90%**. This objective will be reached by reading and understanding the current literature on the subject of person recognition, and choosing between the different frameworks available.

- **Implement a state machine with ROS, so that the robot reacts accordingly to the visitors**. This objective will add usability and modularity of the project, and will allow us to have a functional service robot responding to several inputs given. The robot must be autonomous and react as desired whenever a person rings the bell of the flat. It is also necessary to allow him to communicate, so that it can be used with people who are not familiar with the developed software.

- **Use the developed software in the competitions and achieve the maximum score**. This project is oriented to be used in the European Robotics League service robots competition, so one of our objectives is to finish it before the competitions, and obtain good results when participating in them, adding the time limitation factor to our project.

1.6 Project methodology

The duration of the software development was from July to November 2017. However, the testing period began only in October 2017, due to the fact that the TIAGo robot was provided around that time.

This project was developed at the same time as other pieces of software, developed by other people, which were necessary to participate in the competition. Even though the development of the software addressed in this project

was individual, it was necessary to have weekly team meetings in order to synchronize our achievements and inform the rest of the team which state was the software in.

The individual project methodology was a **Scrum for One** approach. This entails working towards clearly defined and short term goals, constantly self-reflecting on the possible issues in the software, and focusing on the next steps available, instead of trying to tackle the problem on its whole.

The team's project methodology, in preparation for the competition, included weekly team meetings where all team members presented their progress. All questions and requests about the tasks were shared, and every team member needed to provide a list of objectives for the following week. This allowed to maintain a follow-up and to organize which tasks needed more time. This highly iterative approach allowed to provide partial results quickly, and identifying risks as soon as possible.

The evaluation of the success of the project has two parts.

1. We evaluated the success of the software inside the competition, by getting the total number of points obtained in the task, and the overall ranking in comparison with other teams. This evaluation was done by obtaining the results in the competitions in Barcelona and in Edinburgh, and comparing if there was any improvements between them.

2. We evaluated the accuracy of the person recognition algorithm, checking if it was what we expected when defining the first two sub-objectives. This was done some time after the ERL, by using the images recorded by the camera during the competition, due to the fact that, when preparing for the competition there was no time to do explicit tests, and we prioritized achieving the best results as fast as possible. This evaluation of the software was done by creating a ROS node for every specific test, and running it while playing the recorded images. The benefits of doing this was that we were able to recreate the competition situation as accurately as possible.

Throughout the project, it was necessary to use the TIAGo robot for evaluation. This implied creating specific tests for evaluating and using ROS tools so that the analysis could be done faster and independently from other factors. However, whenever the system had to be setup, there were many steps to take, in order to correctly initiate communication between robot and computer and ensure everything was running appropriately. The robot was used by different teammates to do their own tests during the development, and because it has an autonomy of four to five hours it was necessary to organize schedules for each of the persons to use it in specific times, while keeping in mind to charge the robot when nobody was using it.

The programming language used for the project was mainly C++. The ROS framework was also used since it is the one that the TIAGo robot is programmed in. Some improvements were also made by using Python and the

face_recognition package, which provides a simple face recognition functionality and is built on dlib.

All the software developed was stored in the IRI gitlab, always maintaining the master branch as stable, and informing the team-mates whenever big changes had been made.

After the competitions, from January to June, the project was documented into a report, done by distance from Edinburgh, but keeping in touch with the project director (Guillem Alenyà) and supervisor (Javier Vázquez).

1.7 Design decisions

We wanted the developed software to run on a robot, so two decisions were made beforehand in order to have a good result in the competition.

- **Real-time** Our algorithm had to be able to identify the person in real time and we could not afford the cost of waiting in order to get a response. We attempted to give a response in less than 5 seconds, taking into account the time necessary to apply the face recognition algorithms. It would be better to reduce the response time to around 2 seconds.

- **Efficiency** Our algorithm needed to be as efficient as possible, and prevent from recalculating costly structures. The goal was to have all the previously calculated data stored and during the competition test-runs we only performed the essential calculations for person recognition.

Moreover, for each of the decisions taken in the project, different alternatives have been analyzed beforehand, and we selected the one which provided better results. The alternatives devised for each section of the project will be presented in the introduction of each chapter, before explaining the final decision taken.

In general, concerning the whole project, the design includes a basic library, ROS server nodes, ROS modules, and the ROS state machine node. Although this design decision means having many different pieces of software that must be kept updated, it improves the modularity that we want to achieve. If, instead of this, we were to program all the behavior in the same piece of software, any time we wanted to modify it we would need to change a lot more of code. By using this modular design, a small change in the person recognition library did not mean having to change the whole project.

1.8 Laws relevant to the project

In February 2017, the European Parliament debated and voted in favor of regulating the development of artificial intelligence in robotics in the European Union. The draft report for this debate [7] contained proposals designed to address possible ethical issues arising from the boost of robotics production, and included references to Frankenstein, and to Isaac Asimov's laws of robotics.

Specifically, there is a section that explicitly mentions "care robots", which would be the category in which this project falls into, and it points out that human contact is one of the fundamental aspects of human care, and discusses the risk of dehumanization of caring practices by replacing humans with robots. Concerning the researchers in robotics, the document specifies that the following principles should be followed : Beneficence, non-maleficence, autonomy and justice. In this project, we took into account these guidelines when developing the software and always prioritizing the wellbeing of the human that had to interact with the TIAGo robot.

Besides this, there are some privacy concerns that rise from the camera usage, in which we had to make sure that no images were stored without the consent of the people in them. However, our system was used in the controlled environment of the competition, and it will not be used later in a real home. In the competition, all the participants agreed to be recorded and they were aware that the camera was being used, so there are no laws preventing the development of the project tests.

Chapter 2

Planning

In this chapter we will show the initial plan for the development of the project, as well as the final plan. The different deviations from the original planning are also explained in this chapter.

2.1 Project planning and schedule

The project was divided in two time-limited parts.

- **Part 1** The first part was the development part, in which the code for the project was implemented, tested and debugged. It started in July 2017, when the project was announced and the IRI team decided to participate in the ERL Local round. This part had two strict deadlines, which were the starting dates of the competitions; the first one being the 20th November 2017, and the second one the 22d January 2018.
- **Part 2** The second part was the writing part, in which the memory of the project was written and some new data on accuracy was gathered. It started on January 15th 2018, where the outline of the document was defined, and finished on the 25th June 2018.

This division is due to the fact that the code had to be finished before the first robotics competition, and improved for the second. We will now present a specific planned time line, and the later modifications suffered due to various reasons.

2.2 Tasks' descriptions - Development phase

2.2.1 Literature review and documentation

Plan July 2017 - 1 August 2017
Real July 2017 - 1 August 2017

Description The objective of this task was to obtain information before starting the project development. This included reading several papers, books and algorithms about the current approaches for person recognition using computer vision algorithms and machine learning. This task took a long time because of the unfamiliarity with reading published papers and understanding the concepts behind them. Also, it was necessary to access some academic pages from where papers could be downloaded.

2.2.2 Library design

Plan 1 August 2017 - 31 August 2017
Real 1 August 2017 - 31 August 2017
Description The objective of this task was to design a C++ library that could recognize the four different people for the ERL competition. Different person recognition methods were considered, choosing the one which we thought would give better results. For this task, which was only designing, no additional material was needed.

2.2.3 Library implementation

Plan 15 August 2017 - 30 September 2017
Real 15 August 2017 - 30 September 2017
Description This task's goal was to implement the design using C++ and OpenCV, as well as other specific libraries, such as dlib or the face_recognition package for python, which contain useful functions for person recognition. The time period for this task overlapped with the design phase, due to the fact that when starting the implementation, it was likely that it was necessary to revise the design. This task needed a computer with a GPU that had the required processing power and the necessary libraries installed. This initial implementation had to be reconsidered after the first competition, this is explained in section 2.6.1

2.2.4 ROS modules design

Plan 20 September 2017 - 4 October 2017
Real 20 September 2017 - 4 October 2017
Description This task included deciding with the IRI team how the ROS modules would work and specify which requirements were needed by each competition task. Specifically, it was necessary to design the ROS wrapper and ROS module for person recognition library, which had to contain the functions necessary for the task. For this, the expert people in our team explained which was the best way to develop a ROS project of this magnitude and recommended the

best strategy for implementation.

2.2.5 ROS modules implementation

Plan 2 October 2017 - 15 October 2017
Real 2 October 2017 - 15 October 2017
Description This task started once the ROS modules were defined and set, and all specific requirements were stated. Then, using a computer which had ROS installed, the ROS nodes and modules were implemented by using the IRI scripts to create them. These scripts are useful to make sure that all the projects' code follows a homogeneous template, and that all code is followable and understandable. They are also an easy way to get started with ROS programming, which has a steep learning curve. For doing this, it was necessary to ask for some help and guidance to people in the IRI team who have more experience in developing code in ROS.

2.2.6 Task state machine implementation

Plan 15 October 2017 - 30 October 2017
Real 15 October 2017 - 30 October 2017
Description This task needed to implement the state machine for the TBM2, the competition task benchmark that the robot had to complete. It could only be done once the ROS modules were implemented, so that all developed functions from other modules were callable and working. This initial state machine was not definitive, and it was improved during the testing (Section 2.2.8) for the competition 1, where final details were added. However, after the first competition, there was a deviation from the plan and some modification had to be made, explained in Section 2.6.2.

2.2.7 Setup for competition 1

Plan 1 November 2017 -8 November 2017
Real 1 November 2017 -8 November 2017
Description The competition held in Barcelona had some very specific requirements, such as setting up a fixed IP for the computer and for the robot, logging some of the outputs of the task and communicating with the task manager, which was a software that ran on another computer. Due to the fact that we did not want to encounter any problems related to these factors during the competition, we prepared the setup some weeks before, so that during the competition days we could concentrate on other issues. To do this, a static IP was assigned to all devices and the task manager software was executed in an additional computer. Some external help from people who knew more about

network setup was necessary in order to complete this task.

2.2.8 Testing

Plan 8 November 2017 - 21 November 2017
Real 8 November 2017 - 21 November 2017
Description One week before the competition 1 took place, all the tasks needed to be tested in a simulated environment. To do this, all the software had to be finished and ready to be tested, in order to find some of the errors that could still be corrected. To simulate the competition space, two rooms were provided by the Mathematics Faculty at UPC (FME) so that the IRI team could work on the competition and prepare all the tasks. The state machine was improved because this setup allowed to simulate the communication with the electronic devices in the flat, such as the doorbell.

2.2.9 Competition 1

Plan 20 November 2017 - 25 November 2017
Real 20 November 2017 - 25 November 2017
Description The competition was five days long, in which during the first two days there was a testing period, where all participant teams could map the environment, get used to the flat distribution and fix minor mistakes. The las three days were for competing. Each team had a specific time slot to do each task, and there was some extra time at the end to repeat some of the tasks if desired. Each task was done a total of four times, and the final result was the median of all the attempts. In this stage of the project not much of the code was modified, however some new factors such as the competition stress and working as a team were very important during that week. The competition's location was at the PAL Robotics office in Barcelona, so the group of organizers of the competition, all members of PAL, were present during these days.

2.2.10 Improvements for Competition 2

Plan 10 December 2017 - 20 December 2017
Real 10 December 2017 - 19 January 2018
Description Once the competition was over, an evaluation was be made on the results, and it was assessed which of the tasks could be improved before the following competition, at the end of January. Due to the fact that there was not much time, it was necessary to prioritize the tasks that could get better results in this short period.

2.2.11 Competition 2

As mentioned previously, there were two competitions, one in Barcelona and one in Edinburgh. The Edinburgh competition isn't timed in this plan due to the fact that it was announced after the end of the first competition, and I was unable to attend, so my teammates participated in it using the software I developed.

2.3 Tasks' descriptions - Reporting phase

2.3.1 Outlining the project document

Plan 20 December 2017 - 15 January 2018
Real 20 December 2017 - 15 January 2018
Description. As soon as the core development phase ended, the reporting phase started, because it was easier to remember precisely everything developed in the previous months. The outline was designed and then approved by both the supervisor and the tutor, so that during the rest of the reporting phase, it was clear how the project documentation would look like.

2.3.2 Gathering accuracy data in different test sets

Plan 1 April 2018 - 24 April 2018
Real 1 May 2018 - 30 May 2018
Description Due to the fact that the project's development phase was focused on a competition, there was no space for gathering data of the result's accuracy. However, it was agreed that the project report should have a section in which the different person recognition methods were compared with numerical data. This data was gathered and analyzed so that specific numeric results could be presented in this report. However, the original plan was not followed, as is explained in Section 2.6.3.

2.3.3 Writing the document

Plan 28 January 2018 - June 2018
Real 28 January 2018 - June 2018
Description The last phase of the project included writing everything done in a document, and present it. During this phase, there was continuous contact with the tutor and supervisor and updates on how it was progressing. This phase was done on-line from Edinburgh, only needing to access the developed code which was still in the IRI gitlab repository, and occasionally accessing the IRI computers to run some tests.

2.4 Estimated vs actual time

Task	Estimated hours	Actual hours
Literature review and documentation	90	**82**
Library design	50	**63**
Library implementation	90	**98**
ROS modules design	40	**27**
ROS modules implementation	30	**29**
Task state machine implementation	35	**32**
Setup for competition 1	25	**16**
Testing	20	**23**
Competition 1	30	**31**
Improvements for Competition 2	30	**49**
Outlining the project document	15	15
Gathering accuracy data in different test sets	40	**53**
Writing the document	50	**57**
Total	545	575

2.5 Gantt chart

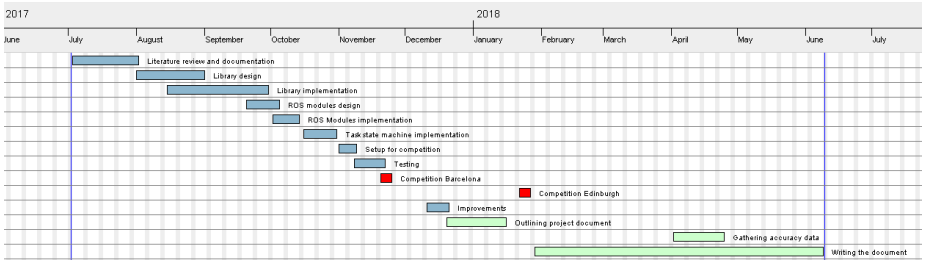


Figure 2.1: Initial plan

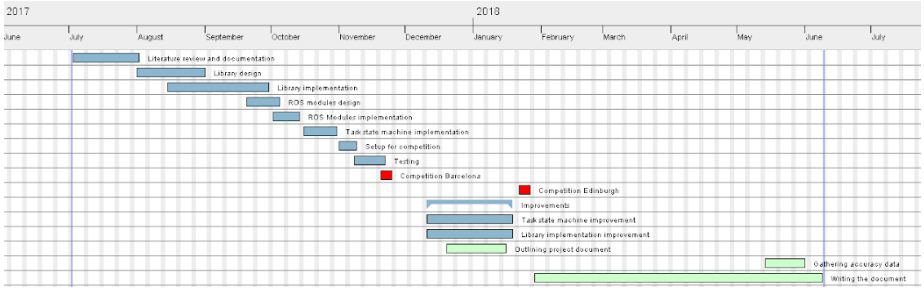


Figure 2.2: Final plan

2.6 Alternatives to the action plan

As mentioned, the development of this project was focused around a competition, which meant that the deadlines were strict. However, as in any project, many things could go wrong, so here are some possible alternatives that were considered.

If any of the mentioned tasks was not completed in the expected amount of time, the first option was to give up on achieving a certain number of points in the competition, and to achieve the maximum score with only a part of the task completed. The alternative was to invest more hours than planned in the harder tasks, delegating some other easier tasks to the rest of the team.

If the team would not have been able to participate in the competition for any reason, the project would have gone on as usual, with fixed deadlines, but it would have been solely focused on obtaining the best results in person recognition, without considering all other competition tasks or other aspects such as the setup.

If some ROS modules were needed which should have been developed by other team members, and they were not implemented in the expected time, team members could rotate tasks and provide help to the others so that everything was ready in time for the competition and nobody had to wait for others to finish their tasks.

Usually the work was done during the weekdays, in which the UPC buildings are open and there is access to the IRI offices. However, during the weeks before the competition it was necessary to work on weekends, in order to do a final sprint.

Between July and December 2017 the dedication for this project was 20 hours per week, while from January to June 2018 it has been reduced to 10 hours per week.

2.6.1 Deviation : Library implementation

When attending the first competition, the results for the person recognition were not good, which although it did not prevent us from achieving a good score, it did limit our maximum possible score. During the improvements phase (Section 2.2.10), we were able to improve this result and obtain a good classification rate before the second competition.

2.6.2 Deviation : Task state machine

During the preparation for the first competition, there was a module that there was no time to implement. Specifically, the expected behavior of the robot when Dr. Kimble is visiting, is to wait for the doctor to leave Granny Annie's room, and then follow him to the door. However, since there was no time to implement this, we set a fixed timer for the robot to wait a certain time, and then proceed to lead the doctor to the door. This issue was then addressed between the first competition and the second one, and solved (see 2.2.10)

2.6.3 Deviation : Gathering accuracy data

Due to different external reasons, such as final exams and great amounts of university work, the accuracy tests (see 2.3.2) were done during the month of May instead of April, which is an unforeseen deviation from the action plan but which was addressed as soon as it was possible.

Chapter 3

People disambiguation and recognition

We will start this Chapter by stating the difference between person detection and recognition. On one hand, when talking about *face detection*, we refer to the algorithms used to select the region of the image that contains a face or a person. On the other hand, when talking about *face recognition*, we refer to distinguishing between the faces of different individuals. We could synthesize these two concepts as understanding **where** the person is in the image, and **who** he or she is.

After briefly introducing the current methods for detection, we will explain the algorithm chosen for person disambiguation in the competition.

Vision algorithms for face detection and recognition have significantly improved in the last years. Nowadays they are used in many places, from surveillance cameras which detect intruders, to phones which use face recognition to be unlocked. The person identification process can be split into two parts; first the detection and then the recognition. The face detection problem has been largely studied in literature [18] and, for the most part, in the case of frontal upright face images, it is considered to be solved. Nevertheless, the problem of recognizing faces is much harder because it depends on lighting, the person's position and expression, the camera's resolution and other aspects such as wearing glasses or having facial hair.

3.1 State of the art for person and face detection

Before discussing some algorithms for face detection, it is important to note the existence of many resources available for the study of face detection algorithms, amongst which, databases of thousands of images that researchers use in order to obtain reliable results in their studies. We will now mention some of these resources.

The FERET database [17] was developed by the Department of Defence

Counterdrug Technology Development Program Office, who sponsored the Face Recognition Technology program. The goal of this program was to sponsor face recognition research, to collect a large database and to perform evaluations of the recognition algorithms. It contains a total of 14,126 images, and a total of 1199 individuals appear in them. These images were taken between August 1993 and July 1996.

Many research groups publish their own datasets. Amongst them, Carnegie Mellon University, Massachusetts Institute of Technology or Harvard University have published their face databases, and many studies test their results on one or more of these databases. As a result, it is sometimes difficult to compare the results of different studies due to the fact that they have been tested on different databases.

The first face detection system described in this section, developed by Rowley et al., [19] is one that is based on neural networks. This algorithm consists of two stages: first, it applies a neural network-based filter to a region of the image which has a size of 20x20 pixels, and whose output is a value between 1 and -1, indicating the likelihood of that region containing an face. This filter is applied at every location in the image, and in order to detect faces of different sizes, the image is scaled and the filter is applied at each size. This first step allows the process to be invariant to scale, lighting and position by applying some preprocessing operations to the image. The faces detected in a sample image with this method is shown in Figure 3.1. In it, we can observe that the true face has many detections at nearby positions and different scale, while false positives have much less detections. Using this information, a heuristic was developed in order to eliminate the false positives and keep only the true face detection. The centroid of the face is calculated by all the nearby detections. The results of this system is detecting between 78.9% and 90.5%, with some number of false detections.

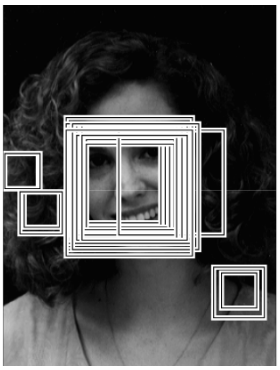


Figure 3.1: Regions of the image which are identified as faces with Rowley's algorithm. Source ; [19]

The next detection system relevant to this project is based on the Sparse Network of Winnows (SNoW) learning architecture [12]. This is a network which

consists of linear units over a feature space, and are connected between them through weighted edges. The output of this network has only two options: *face* and *non-face*. Each input image is mapped into a set of features which appear in it, then, this representation is presented to the input layer of SNoW, and propagates through the linear units. A linear unit will become active if the sum of weighted edges of all of the features that are present is above a threshold. This method has obtained results of up to 94.8% detection rate.

The most commonly used method for face detection is the Viola-Jones algorithm [23]. Their strategy is to create an "integral image", which is an intermediate representation of the image. Specifically, the integral image at pixel (x,y) contains the sum of the pixels above and to the left of (x,y), inclusive. Once this is built, they used a variant of AdaBoost to train a classifier and to select the most relevant features. Adaboost is a learning algorithm, and by using it, they obtained the main rectangular features that define a person's face, shown in Figure 3.2. The final step is to run the image through a cascade of classifiers, which increases the detection performance. Their results showed 93.7% of accuracy, with a lower rate of false detections on the MIT-CMU dataset.

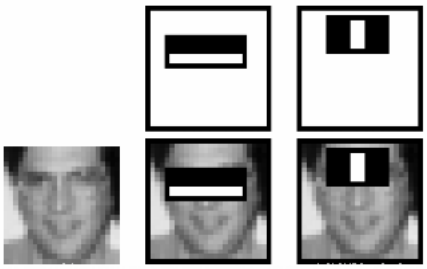


Figure 3.2: The features selected by AdaBoost. Source : [23]

After the great impact of this research, many other attempts have been made to achieve an efficient person recognition algorithm. Amongst them is the Histogram of Oriented Gradients (HOG) for human detection [6], which achieved near-perfect results in the MIT pedestrian database. The methodology consists of dividing the image in different regions, and for each of them, computing the horizontal and vertical gradients in it. Then, quantize the gradient orientation in 9 different bins in order to create a histogram. Their results showed a miss rate (that is, false negatives over the sum of true positives and false negatives) of 0.01 on the MIT Pedestrian dataset.

The most common framework used for image processing is OpenCV, which is a library of functions aimed at real-time computer vision. It is an open source library, launched in 1999, which supports C++, Python and Java programming languages. This library includes implementation for the last two algorithms described in this section, and we will use it in order to compare which one gives better results for the task at hand.

During this literature review phase, while reading about different person de-

tection and recognition experiments, it was observed that most of them followed the same methodology:

1. Detect the part of the image that is the face.

2. Extract descriptors from the face.

3. Compare these descriptors with a database of previously stored images.

We decided to follow this design methodology when trying to recognize the people, but we explored several alternatives, which we report in this chapter. For the face detection part, we analyzed the results of applying a Histogram of Oriented Gradients [6] to the image, compared to using the Viola-Jones approach[23]. For the feature extraction and comparison, we tried two different methods; firstly using the dlib framework [15] and secondly using the Python library of face_recognition[10]. The comparison of these methods is shown in Chapter 6.

3.2 Disambiguation through appearance

After understanding the current state of vision algorithms, and establishing the design decisions taken, in this section we introduce an algorithm for recognition of the visitors. Keeping in mind the four possible visitors and their appearance (shown in Table 3.1),we will develop an algorithm to be able to distinguish between them. The Postman and Deli Man have specific uniforms (which are shown in Figure 3.3). These images are provided in the official ERL rules, given so that the participants know what the visitors will look like months before the competition.

Visitor	Shirt color	Face
Deli Man	White	Unknown
Postman	Yellow	Unknown
Dr.Kimble	Any	Dr. Kimble's face
Unknown	Any	Unknown

Table 3.1: Differences between visitors

In this case, we will first try to detect the shirt location to find the shirt color and check if it is either the Deli Man or the Postman. Then, we will try to recognize the face to check if it is Dr. Kimble and the last case would be that

(a) Mockup uniform of the Deli Man (b) Mockup uniform of the Postman

Figure 3.3: Different uniforms of the visitors

it is unknown. A high-level description of this process is shown in Algorithm 1

```
Data: Image from camera
Result: Person
initialization;
shirtPosition = detectShirtPosition(Image);
shirtColor = detectColor (shirtPosition);
if shirtColor=='YELLOW' then
    Person = POSTMAN;
else
    if shirtColor=='WHITE' then
        Person = DELIMAN;
    else
        face = recogniseFace(Image);
        if face == 'DRFACE' then
            Person = DRKIMBLE;
        else
            Person = UNKNOWN;
        end
    end
end
```

Algorithm 1: High level disambiguation algorithm

As shown in Algorithm 1, the initial step is to identify the shirt position and color, in order to distinguish between the Postman and the Deli Man. If the shirt color is neither yellow nor white, then a face recognition algorithm must take place in order to recognize if the person is the doctor or if he is unknown.

The first step to recognizing the shirt position, is locating the eyes or the face, by using face detection algorithms. Then, use that information in order to locate the body. Once the main body region is detected, apply a color clustering algorithm to find which color it is.

3.3 Face detection

In order to detect the faces, the cascade classifier algorithm is one with the best results. OpenCV provides a module named CascadeClassifier, which is a detector inspired by the one described by Paul Viola [23], and provides support for both object detection and face detection. In order to initialize this class, we used a pre-trained haar classifier cascade provided by OpenCV, which was stored in an XML file. The file used was a face-detection cascade classifier.

Once we loaded the file and the pre-trained classifier, we could use the functions provided by OpenCV to detect people's faces. Specifically, we used the function `detectMultiscale`, which uses one of the cascade classifiers for face recognition to detect all faces in the image. This function returns a vector of all the faces detected in the image, defined by a `Rect` structure, which contains the x and y coordinates of the top-left corner of the face, the width and the height.

In this scenario, we only wanted to find one person in each image. This is why, whenever more people were found, an error was returned in order not to analyze the image further, because it would be impossible to distinguish which of the faces is really the one of the visitor and which isn't.

An alternative for face detection is to use the Histogram of Oriented Gradients. As well as with haar classifiers, OpenCV provides an implementation of this algorithm, which is a module named HOGDescriptor. This class also provides the function `detectMultiscale`, which finds all the people in the image by applying Dalas' algorithm[6], and returns a `Rect` which indicates the position of the person's body in the image.

We then compared the detections and selected the Viola-Jones algorithm, as it provided better results (see Section 6.1 for details).

3.4 Body detection

By choosing the Viola-Jones method, once the face detection part was finished, we had obtained a Rect structure with the person's face, and the next thing to do was to find the person's body, so that we could find its shirt color. In order to do this, we had to set the values of a new Rect structure, that is, x and y coordinates which identify the upper-left corner, width and height of the body. Recall that we identify the person's face with the following structures: Face.X, Face.Y, Face.W, Face.H. The full image has R rows and C columns. With this, the next formulas show how the calculations were done to find the person's body. Figure 3.4 shows a sketch of what these calculations look like in an image.

$$Body.X = max \begin{cases} Face.X - \frac{Face.W}{2} \\ 0 \end{cases}$$

$$Body.Y = Face.Y + Face.H$$

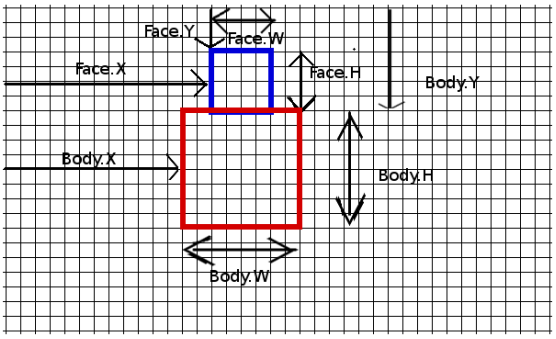


Figure 3.4: Body Position relative to the face position

$$Body.W = min \begin{cases} 2 * Face.W \\ C - Body.X \end{cases}$$

$$Body.H = min \begin{cases} 2 * Face.H \\ R - Body.Y \end{cases}$$

This means, that the person's body upper left corner (Body.X, Body.Y) will be, for the X coordinate, subtract half the face's width from its left corner's X position. For the Y coordinate, add the face's height from its left corner's Y position. In the first case, if the obtained value is negative, then the value will be zero. However, for the Y coordinate we can assure that it will be inside the image, because it is the same as the lower corner of the face rectangle, which will always be inside the image, due to the characteristics of the function explained in the previous section.

For the width and height, it will be twice the face's width and height. However, if this means that the body does not fit in the image, then the body's measurements will be the maximum allowed that will fit the image.

3.5 Color clustering

The next step is to check which color the person's shirt is. At this point, we have the image from the camera, and a Rect structure specifying the person's body position. We can then select only the body region and work with that sub-image.

Firstly, we will use a median filter to blur the image, so that the shirt region has a much more smooth color. Next, we will find the clusters using the K-means algorithm. In order to use it, we need to reshape the image so that we have a list of (R,G,B), instead of a matrix. To do this, we will create a new

structure which will have length of the rows times the columns of the image, and will have a width of 3, for the three color channels. In Figure 3.5a is the matrix representation of the image, as we obtain it from the camera. Figure 3.5b shows the resulting structure in order to correctly apply k-means.

(a) Image as a matrix (b) Image as rows of colors

Figure 3.5: Different ways to store image information

K-means is one of the most popular clustering algorithms. Its strategy is to store centroids for each of the data groups. In order to find the best centroids, the algorithm alternates between assigning data points to the different clusters, and choosing centroids based on the current clusters.

These two steps are repeated until either the algorithm converges, or a maximum number of iterations is reached. The convergence is measured by checking the accuracy epsilon.

In the K-means algorithm, we specify the ending criteria, which in this case was that the algorithm did 10 iterations or that the distance that the centroids moved in the last iteration is less than 1.

We also had to specify how many clusters we wanted to find. In our case, we set it to 5 clusters. This was big enough so that the true color of the shirt was found, but small enough so that there was not much noise in the resulting clusters.

Finally, we specified which method we wanted to use to find the starting points of the centroids, which is the one Artur and Vassilvitskii specified in 2007 [2]. They designed a randomized seeding technique in order to obtain an algorithm competitive with the optimal clustering, improving both speed and accuracy of the k-means algorithm.

The kmeans function returned a data structure with the cluster index assigned to each of the image points. Using this, we iterated it searching for the biggest cluster, that is, the one that had more pixels assigned to it. The color of that pixel was the one of the person's shirt.

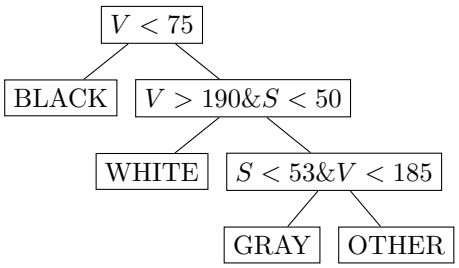


Figure 3.6: Decision tree to find the color

Figure 3.7: Intervals assigned for each value of Hue

3.6 Find color

The last step was to convert the color value to one of the basic colors known. To do this, we converted it to HSV and classified it into 11 possible color categories depending on the Hue, Saturation and Value, which range between 0 - 180, 0 - 255 and 0 - 255, respectively. These values are specific for the OpenCV libraries which are the ones we used, but other representations are much more common, such as having Saturation and Value range from 0 to 100 and Hue from 0 to 360. The possible colors were black, gray, white, red, orange, yellow, green, aqua, blue, purple and pink. The classification was done following a sequence of conditions.

Firstly, concerning the Value (V) and Saturation (S), we classified between black, white and gray. If Value was less than 75, it meant the color was very dark, so we assumed it was black. If Value was higher than 190 and Saturation was less than 50, it meant that the color was bright but unsaturated, which was likely to be white. Finally, if Saturation was low but value is high, we assigned the color gray. Otherwise, analyzing the Hue, we found between which of the other colors it was. The parameters shown in Figure 3.6 were tuned by hand in order to obtain the best results. The values that marked the different color intervals were based on the usual representations of color and common value assignation.

3.7 Face recognition

Once the face was detected, the body located and the shirt color identified, we proceeded to recognize the person's face. Before starting, we reviewed the current techniques used for face recognition.

3.7.1 State of the art for face recognition

Face recognition has become much more popular in the recent years, because it is a non intrusive technique for identification.

In order to test the performance of face recognition algorithms, extensive databases are available. Amongst famous data sets available there is the Yale face database, which contains 5760 images of 10 subjects under 576 different viewing conditions for pose and illumination. Another relevant database is the Labeled Faces in the Wild (LFW) [13], which was designed to study the problem of unconstrained face recognition. It contains over 13,000 images of faces obtained online, with the faces labeled with the name of the person in it. The only constraint on the faces in this dataset is that they were detected by the Viola-Jones algorithm, mentioned in the previous section.

A good face recognition system must meet the following requirements[16].

1. The speed of the system must be within a threshold.
2. It should have a high accuracy.
3. The system should be easily enlarged, so that it can recognize a higher number of subjects.

Face recognition algorithms can be divided into three categories:

- Holistic matching methods, where the whole image region is taken as input for the system. This category includes Eigenfaces, Principal component analysis and linear discriminant analysis.
- Feature-based methods, in which local features (such as eyes or mouth) are extracted and passed through a structural classifier.
- Hybrid methods, which combine the two previous methodologies, and are mostly used with faces in 3D.

The first algorithm we will discuss here is the Eigenface method[22], which falls into the holistic category, and is based on Principal Component Analysis. This system was developed in 1991, and is based on the following process. Firstly, the images are inserted into a database. Secondly, the characteristic features are extracted from each face, and they are represented as a vector, thus creating the eigenfaces. Then a normalization of the input image is done, so that mouth and eyes are aligned with the database images, and they all have the same size. Finally, the eigenfaces are extracted by using Principal Component Analysis (PCA), which projects the face images onto a feature space of less dimensions, that spans the relevant variation among known face images. These significant faces are the eigenvectors of the set of faces (thus the name Eigenface). Each image is then represented as a vector of weights, and in order to recognize a particular face, the image's weights are compared to those of known individuals. The flowchart of this algorithm is shown in Figure 3.8. Their results showed that they obtained up to 96% accuracy when varying lighting, 85%

when varying orientation, but when the size was changed, accuracy dropped to 64%.

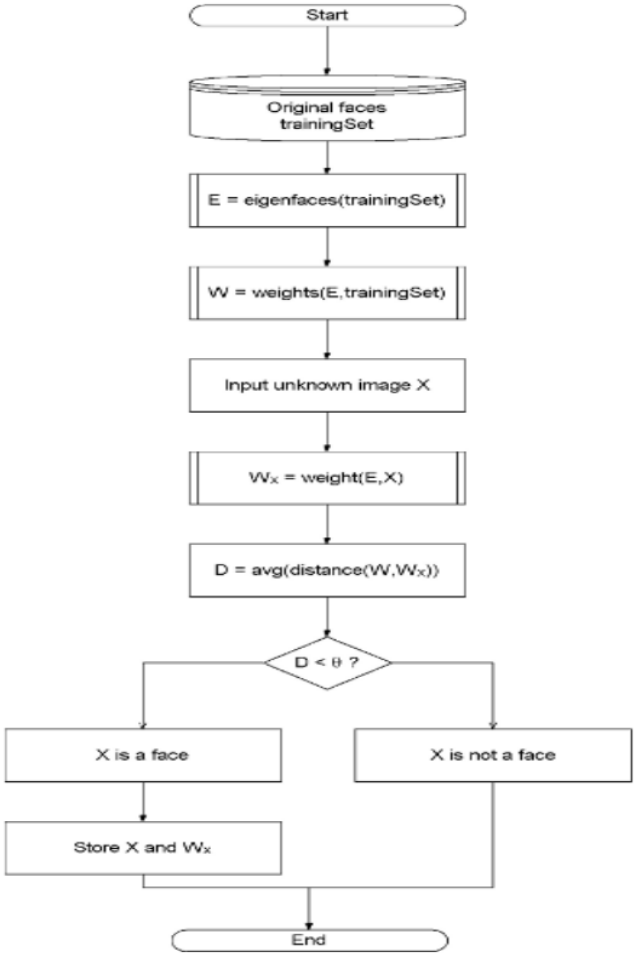


Figure 3.8: The flow of the Eigenface algorithm. Source : [16]

The next relevant algorithm is the Fisherface method [4], which obtained lower error rates than Eingenface technique, by using PCA combined with Fisher's Linear Discriminant Analysis. The method is the same of projecting the image into a subspace, which will discount the regions of the face with large deviation. This creates separated classes in a low-dimensional subspace, even in adversarial conditions such as a variation in lighting and change in facial expressions. The goal of this method was to maximize the ratio of between-class scatter versus the within-class scatter, that is, that the samples of images belonging to the same individual are close to each other, while the images of

different people are far, in the subspace in which the image is projected. In their study, they compared their results with the ones obtained by applying the Eigenface method. They found that the error rate was 7.3% when Eigenface's error rate was of 24.4%. Additionally, they attempted to recognize people who wore glasses, with an error rate of only 5.3%.

Due to the fact that the largest application of face recognition techniques is surveillance and security, recognizing people from video sequences has become more and more relevant[14]. These algorithms can be decomposed in three parts: detecting the face, tracking it, and recognizing it. Usually these methods select a few frames from the camera and apply basic face recognition techniques to them.

Many studies are appearing recently related to face detection from 3D images. The advantages of this are that it allows to detect new features such as the curvature of the face while maintaining lighting invariance. The methodologies used to obtain 3D images ranges from scanning systems, which capture directly the depth coordinates of the face, to stereo vision systems, which extract 3D information from two or more 2D images from different viewpoints. When dealing with 3D faces, the hybrid method is used[16], which consists of five steps: *Detection*, in which the region of the image containing the face is detected, *Position*, in which the head's location and orientation is determined, *Measurement*, in which each curve of the face has specific measurements assigned in order to make a template, *Representation*, in which this template is converted into a numerical representation, and finally, *Matching*, in which the obtained data is compared with the stored database.

Analyzing these algorithms, a new question arises, which is whether a face recognition system could be able to distinguish between a person's face and a photograph of this person. This loophole could suggest a great security threat which should be addressed. This was analyzed by Cho et al. in 2016 [5], when the face recognition methods were tested and compared for live faces and high definition face videos from a LED display. In it, they tested three face recognition engines used for commercial purposes. The results obtained, shown in Figure 3.9, were that, indeed, face recognition systems could be tricked with video images of the person they were attempting to identify, when tested in homogeneous lighting conditions. Although the recognition rate was only slightly smaller when the images were fake, it still presents a risk when using face identification for security purposes.

Engine	Detection rate(%)			Recognition rate(%)		
	Real faces	Fake faces	Deviations	Real faces	Fake faces	Deviations
A	98.15	97.77	0.38	97.47	94.02	3.45
B	80.86	88.48	-7.62	73.80	75.13	-1.33
C	51.69	46.65	5.04	58.79	53.57	5.22

Figure 3.9: Results of comparing face recognition engines with live faces and videos of faces. Source: [5]

With these methods, the current status is that in cooperative scenarios, when the person is willing to look at the camera with a neutral face and a good resolution, the results are quite satisfying. However, in non-cooperative scenarios, artificial neural nets provide much better results [1].

The final algorithm to talk discuss in this section was developed recently, in 2014, by the researchers at Facebook AI research group, who introduced the DeepFace method [21], achieving an accuracy of 97.35% on the LFW dataset. This method starts by aligning the face in 2D, and then representing the face by using a 3D face modeling technique. Once this alignment is done, the model is rotated to obtain a frontalization, and finally it is ran through a 9-layer deep neural network. This is the face recognition algorithm which is closer to human-level performance, which is of 97.5% .

OpenCV provides some of these face recognition algorithms' implementation, but not all of them. In order to have a greater success rate in person recognition, other frameworks which are designed to implement neural nets are much more useful. Amongst them, dlib[15], which is an open source C++ toolkit for machine learning, and face_recognition[10], which is a python library.

For this project, two different face recognition strategies were used. Both of them included the use of machine learning and neural networks, but had different implementations and results. The first one, used for the competition in November, was developed with the Dlib library, in C++. The second one, used for the competition in January, was developed with the face_recognition library in Python and with a C++ wrapper.

In both methods, there is an initial section in which we store the training data, which are the images of the faces of the known people that we already have, and we use them as a gold standard in order to compare it with the new faces obtained. There is another section in which a new face is classified, by using information from the other known people. In this step, we only use the region of the image which contains the face, since the rest of it does not provide any additional information.

3.7.2 Method 1 : Dlib

Dlib is a C++ toolkit that includes machine learning algorithms and tools that can be used in order to create software in C++. It is open source and available on Github.

The network built to classify the faces is a ResNet [11]. This is a recently developed network which tackles the problem of training deep neural nets. ResNet uses a residual learning framework, which improves accuracy of some of the nets which have many layers. It includes a building block, shown in Figure 3.10. This means taking previously existing neural nets, and convert each pair of two layers to the ResNet, meaning adding the initial values (before passing the two layers) with the values obtained after traversing the two layers.

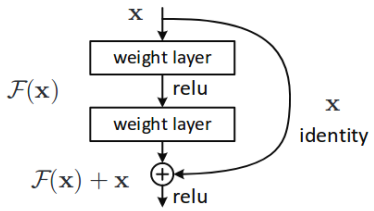


Figure 3.10: ResNet building block. Source : [11]

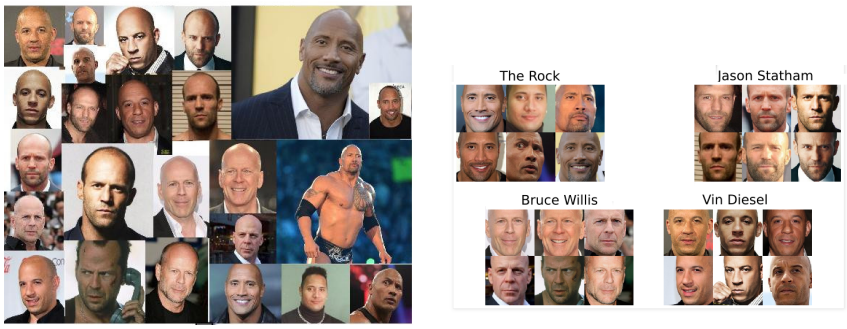
The initial phase is to load the images of the known people. These files are all stored in the same folder, so we load the images and extract the face region, using the face detection algorithm explained in Section 3.3. Each of the images has a label which specifies which person it is. Then, for each of the face regions, we run them through the ResNet to extract its descriptors and store them for using in the following step.

Once the network was trained, it could be stored as a binary file and loaded each time it was needed, in order to save computing time.

Then, having the image of the face region we want to classify, we run the image through the trained network, and obtain the descriptors of the image. Once we have them, we measure the euclidean distance from these descriptors to the ones from the images we used in the training step. When this distance is lower than a certain threshold, it means we have found a match, and return the label of the person who looks the most like the one we want to classify.

One problem we ran into when testing the dlib toolkit was that it included its own image reading functions and data structures, while we were used to the OpenCV methodology, which was what we had used for face detection. We discovered that while the OpenCV handles images in the Blue-Green-Red order, the dlib library does it in an inverse order, that is Red-Green-Blue. This problem implied that we needed to decide to use one or the other functionalities for image manipulation, but not a combination of both.

Dlib provides an example of face recognition with deep learning[15], in which using a trained network, different people are identified in an image. The method used there takes a set of faces of different celebrities and then creates cluster to show which faces belong to the same people, and which of them are different. This example is shown in Figure 3.11. With this algorithm, they had achieved a 99.13% in the Labeled Faces in the Wild benchmark.



(a) Initial image from which all faces are extracted

(b) Clusters obtained by using the dlib algorithm

Figure 3.11: Dlib face recognition example. Source : [15]

Seeing the good results it provided, we decided to use this framework for the first competition. We implemented the library and prepared to collect images to re-train the network once we were at the competition venue. However, we discovered that when using the images from the camera in the competition, the results were not good. The resolution used in the previous example was higher than the one provided by the camera in the competition, and we had to work with a camera that was further away from the subject than in the previous example.

The specific results obtained in the competition are shown in Section 6.2.

3.7.3 Method 2 : face_recognition package

Seeing that during the first competition we didn't obtain good results, we had to chose a new algorithm to implement in order to improve our recognition accuracy.

The second method of face recognition entails using a package provided by Python. This library has a much simpler interface than the previous method, and also uses dlib in the lower level implementation. It was built with deep learning in 2017 and achieved a 99.38% accuracy in the Labeled Faces in the Wild benchmark. This package not only allows to recognize faces, but also to detect them, so when using this functions we will not use the previously explained face detection methods, such as HOG or Haarcascades.

The package includes different functionalities, amongst which loading image files, detecting face landmarks, extracting a set of face encodings and comparing sets of encodings in order to determine if two faces are the same.

The method used for person recognition was the following: First, we took a frontal face image of Dr. Kimble and stored it in a folder. Then, we extracted the face encodings, that is, a 128-dimension vector that encodes the detected face. Then, for each new frame that the camera receives, we will also extract the face encodings and compare the two, by measuring the euclidean distance

between them. Then, if the obtained value is lower than a threshold, which we set to 0.6, the output is that the faces are of the same person. Otherwise, the faces belong to two different people.

This method only compares one-to-one images, instead of storing a database of many different faces as the previous method did. This has the benefit of not needing much storage space in order to run this recognition algorithm.

This approach used the Python programming language. However, all the rest of the software developed was in C++, and since we wanted to homogenize all the developed code, and make it into a library which could be compiled, we needed to wrap the python script with C++. This was done by developing code which called the python functions when necessary, with the appropriate arguments, and correctly extracting the returned values, as well as the exceptions that the python script would throw.

Chapter 4

Module development in ROS

In this chapter we will explain how we integrated the developed library with ROS. ROS presents a change of paradigm from having a single module which includes all of the robot's functionalities, to a distributed structure which allows accessing different modules and simplifies the communication between them.

Until recently, many robotics manufacturers had developed their own programming language and framework. This has been one of the main problems in industrial robotics due to the fact that the lack of unification has made it impossible to easily port software from one robot to another, and adds the complexity of needing to understand every communication protocol and robotic interface before being able to begin working on a robot. This great inefficiency began to be addressed in the mid-2000s, and the idea of unifying all robots under the same open source framework has started to be more popular. However, many manufacturers are unsure of the positive impact of this idea, due to the skepticism surrounding open source software. Currently, one of the most used open frameworks is the Robotic Operating System (ROS), developed in Stanford in 2007 [9]. Since then, its usage has grown around the world, being present in many popular robots such as PR2[1] and TIAGo, and some companies such as ABB[2] and more recently, in 2017, KUKA[3] have started to develop the ROS-specific packages for their robots.

An alternative to ROS is YARP[4], which provides the possibility to build a robot control system as a group of programs which communicate in a peer-to-peer way. They provide support for the different connection types (tcp, udp or local amongst others) in order to match the developers' needs, as well as

[1] PR2 is a personal robot developed by Willow Garage `http://www.willowgarage.com/pages/pr2/overview`

[2] ABB (ASEA Brown Boveri) is a Swedish-Swiss multinational corporation `https://new.abb.com/es`

[3] KUKA is a German manufacturer of industrial robots `https://www.kuka.com/en-de`

[4] Yet Another Robot Platform `http://www.yarp.it/`

a flexible hardware interface. Their goal is to increase the durability of robot software projects.

YARP uses ACE[5] to support extra protocols. ACE is an open-source object-oriented framework which provides implementation of patterns for concurrent communication software. It is developed in C++, and it is targeted for developers who require a real-time communication in the applications they develop.

4.1 ROS Introduction

The Robot Operating System (ROS) is a very helpful tool to develop software for robots. It is a framework that encourages collaborative robotics software development, inspiring groups to build upon each other's work. One of the advantages of using ROS is its distributed and modular design, which allows users to use only the parts of it that are of use to them.[9]

There is a collection of nodes and programs, named roscore, that is required in order to correctly run ROS and to ensure the correct communication between all the modules. In order to start, it is necessary to type roscore in a terminal and this will initialize all the necessary basic structures that are used in ROS.

ROS uses a modular structure, which allows different nodes within the program to interact and communicate with each other, by sending messages. Its structure represents a change of paradigm, from having only one block that includes all the functionalities of the robot, to a set of independent structures, each for a different purpose. This modular structure is shown in Figure 4.1.

[5]The ADAPTIVE Communication Environment `https://www.cse.wustl.edu/~schmidt/ACE.html`

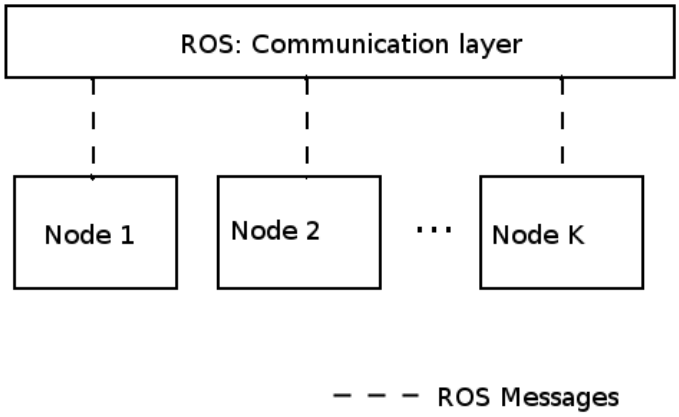


Figure 4.1: ROS as a distributed structure for communication between nodes

ROS includes different messages to communicate between the nodes, which all go through the communication layer. These messages can either be in the form of topics, actions or services; all of them having different programming approaches and guidelines for interaction. The following list explains briefly what each of the functionalities consists of.

1. **Topics**. These are the messages that are continuously published and to which a node can subscribe. Some examples of topics are the robot's position, speed and the images it sends from the camera.

2. **Services**. These are blocking messages that must be requested before being executed. A client calls a server which replies with the corresponding message, or an error if something went wrong. An example of service is requesting to classify an image.

3. **Actions**. These are non-blocking function calls which are requested by a client and they give continuous feedback while the action is being executed. An example of this is navigation or text to speech.

This project operates with ROS because the software included in the TIAGo robot requires it. This means that for each action, service or topic that is needed, a whole ROS structure of message handling must be developed. This introduces a new issue which is that the complexity of our algorithm will be much higher, making it difficult to have a clear and structured code. In order to solve this, the IRI team decided to work with ROS modules, which are the same as an API.

These modules will simplify our code by allowing to encapsulate the complexity of message handling within functions, so that the main code can interact with them without taking into account how it is internally managed.

The following list explains briefly some of the modules used, developed by other members of the team.

1. **Navigation module**. This module contains the functionalities to make the robot move. The robot has a map of the flat stored, in which different way-points and Points of Interest (POI) can be specified, like concrete positions in rooms. There is a function in this module, named `go_to_poi`, which makes the robot move until it reaches the POI. This function is internally implemented as an action and returns `true` when the robot has reached its destination. This is the basic navigation function used in the algorithm for implementing the task.

2. **Text to speech module**. This module allows the robot to say sentences, which is something that is necessary in order for the person who interacts with it to receive feedback. The robot will have to explicitly tell the visitors to come into the flat or to stay away; this is why a text to speech module is used. This ROS module contains a function, named `say`, which is also internally implemented as an action, and will return `true` when finished.

3. **Head module**. This module allows the movement of the head of the TIAGo robot. It is useful when we want the robot to look at a specific point in the room.

4. **Image difference module**. This module does a background subtraction of the current image, and checks whether it has changed. Specifically, in the task, the robot must wait for Dr. Kimble to leave the room, so this was implemented by using this module, which has a function named `has_changed`, which returns true when the image seen by the robot is very different from the background, which indicates that there has been movement.

5. **Devices module**. This module allows the robot to interact with the electronic devices in the house. As mentioned, the flat is a smart home, in which the lighting, blinds and doorbell are connected through a network. This device module contains a function named `listen_bell` which is internally implemented as a topic, and will return `true` if some visitor has rung the bell.

6. **Log and referee module**. These modules are specifically for the competition; the Log module allows to record to a text file every thing that the robot perceives and does, which is necessary for the scoring in the competition. The referee module is the ROS module through which the robot is notified when the task should start, and must be used in order to compete.

Lastly, here is an explanation of how the person recognition library was used to create a new ROS Module.

4.2 Building a ROS Module for person recognition

Once the person recognition library was finished, it was necessary to create a ROS Module. To do so, an intermediate step was necessary which was creating a ROS node, through which a service to classify could be sent. The interaction explained in this section is shown in Figure 4.2.

In this case, it was preferable to use a service rather than an action or topic because we wanted it to be a blocking functionality and it did not need to give any feedback. This node also was subscribed to the camera topic, the stream of images received from the flat's front door camera. Whenever the service named `classify_current_person` is called, the node waits to get a valid image from the camera, and then calls the library function by using this image to classify the person. This service call returns a message which is of the type `classify`, which is a message we defined for this task and its attributes are the following: It contains a string with the label of the classification result, as well as the accuracy for this label, a boolean which indicates the success of the classification and an error message.

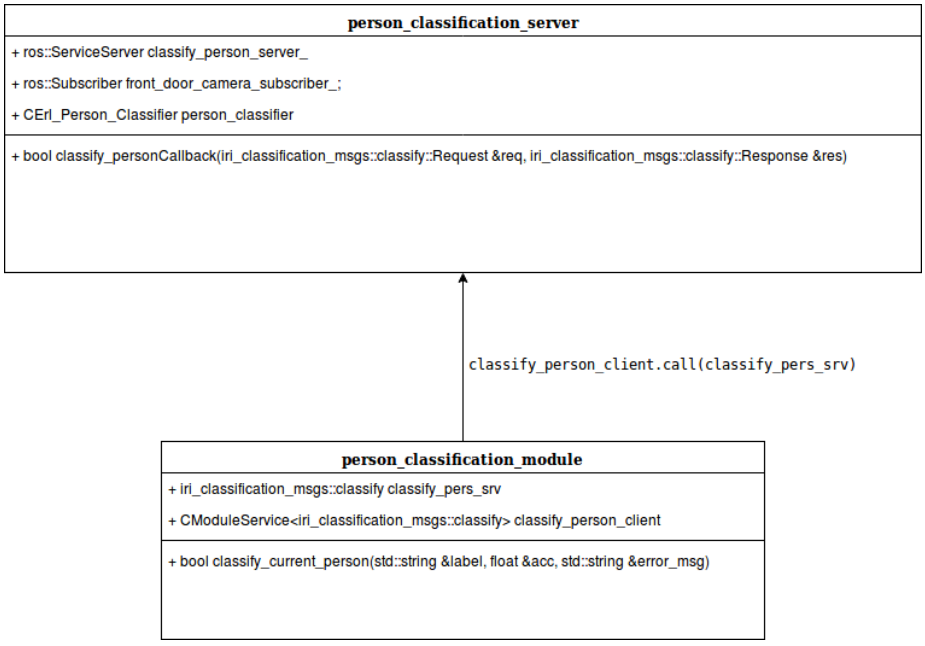


Figure 4.2: Communication in ROS using a server and a module

Once this intermediate node is built, the module can be created. This module only needs one function which is named `classify_current_person`. This function will have three parameters, passed by reference so that the caller can know about the result. These parameters are the label, which is string pointer, the accuracy, which is a float pointer, and an error message which is also a string pointer. The function will return true if everything is successful, and it will contain the person's label and accuracy of the prediction. On the contrary, if there is some error, the message string will explain the problem.

4.3 Implementation

In this section we will present the code from the specific functions used when developing the ROS Server and module for person classification. For the ROS Server, we show the function named `Classify_personCallback`, which is called whenever another node calls the server. As seen in Listing 4.4, we first lock the shared variables so that they aren't accessed simultaneously. Then, we check if everything is loaded correctly; that is, if the image is valid, if the neural net was loaded correctly and if the image is new, so that we don't classify twice the same frame. Then, we call the function in our library that returns the person by accessing the current image previously stored from the camera. Once this function is called, the results are copied into the `res` structure. The final step is to unlock everything that was previously locked, and return true.

The module implementation has a function named `classify_current_person`, which will be called from the state machine whenever we want to classify a visitor. This is shown in Listing 4.4, where a ROS client calls a ROS server. The return value specifies whether the call was successful (`ACT_SRV_SUCCESS`), if it still hasn't finished (`ACT_SRV_PENDING`) or if it has failed (`ACT_SRV_FAIL`). In any of the three cases, the error messages are set accordingly and the result is returned, by accessing the `response` field of the `classify_person_srv` variable.

4.4 Tools

ROS provides several tools which make the process of developing software in robots much easier. The tools mentioned in this section were used during the development of the project due to the fact that they allowed to work more efficiently.

- **Dynamic reconfigure:**A useful tool in ROS is the dynamic reconfigure node. It is a way to show a subset of a node's parameters for configuration. It contains a useful GUI in which different options can be activated and tested independently. For example, in order to test the face recognition functions on the robot, we added a boolean parameter which, when true, called the `classify_current_person` function.
- **Rosbag:** ROS provides a structure named rosbag, which allows to record and play all the topics which have been published during a certain period

of time. This is a very useful tool because it allows to store all the topics that a certain node is subscribed to, and later simulate the exact same situation even if the nodes that publish these messages aren't active.

Listing 4.1: Implementation of **classify_personCallback** function

```
bool ErlPersonClassifierAlgNode::classify_personCallback(
    iri_classification_msgs::classify::Request &req,
    iri_classification_msgs::classify::Response &res)
{

  this->alg_.lock();
  this->classify_person_mutex_enter();


  if (this->has_valid_img && this->has_valid_net && this
      ->is_new_img){
    cv::Mat img = this->current_img;
    std::string error ="";
    Person person;
    float acc;
    bool resultClassify = this->alg_.get_person (img,
        person,acc,error);
    res.accuracy = acc;
    res.error_msg = error;
    res.success = resultClassify;


    this->is_new_img = false;
    switch (person) {
              case Kimble:
                res.label = "Kimble";
                break;
              case Deliman:
                res.label = "Deliman";
                break;
              case Postman:
                res.label = "Postman";
                break;
              default:
                res.label = "Unknown";
                break;
    }
  }
  else {
    res.accuracy = 0;
```

```
    res.success = false;
    res.label="";
    if (!this->has_valid_img)res.error_msg = "No valid
        image from camera";
    else res.error_msg = "No valid net loaded. Try
        calling load_net_attributes and set_cascade_file";
  }
  this->classify_person_mutex_exit();
  this->alg_.unlock();
  return true;
}
```

Listing 4.2: Implementation of **classify_current_person** function

```
bool CPersonClassificationModule::classify_current_person
    (std::string &label, float &acc, std::string &
    error_msg){
  act_srv_status status;
  bool return_state=false;

  status = this->classify_person_client.call(this->
      classify_pers_srv);
  switch (status){
      case ACT_SRV_SUCCESS:

          if (this->classify_pers_srv.response.success){
            label = this->classify_pers_srv.response.
                label;
            acc = this->classify_pers_srv.response.
                accuracy;
            error_msg = this->classify_pers_srv.response.
                error_msg;
            return_state = true;
          }
          else {
            if (this->classify_pers_srv.response.
                error_msg==""){
                error_msg = "ClassificationModule:
                    failed to classify person";
            }
            else {
                error_msg = classify_pers_srv.response.
                    error_msg;
```

```
            }
            return_state = false;
          }
          break;

      case ACT_SRV_PENDING:

          error_msg="pending";
          return_state = false;
          break;

       case ACT_SRV_FAIL:

          error_msg="fail";
          return_state = false;
          break;

  }
  return return_state;
}
```

Chapter 5

Welcoming visitors : State Machine

Now that we have created a ROS module that handles person recognition, we can use this and many other modules in order to develop a state machine for the robot. It will be developed in ROS, and include the actions done in each state and the conditions for transition between states.

5.1 Designing the state machine

When designing the robot behavior for the task, there was the option of designing a state machine and specifying the behavior for each state and the conditions for transitioning to the new state. Using this option increased the readability of the code in comparison to other implementations.

As mentioned before, the Welcoming Visitors task of the ERL consisted of several visitors arriving at the door. When each of them rang the door bell, the robot had to do a specific action.

1. When Dr. Kimble arrives, the robot lets him in, guides him to the bedroom, waits for him to leave and follows him to the door.
2. When the Deli Man arrives, the robot lets him in, guides him to the kitchen to deliver breakfast, and guides him to the door.
3. When the Postman arrives, the robot lets him in, asks him to deliver the package, and guides him to the door.
4. When an Unknown person arrives, the robot must not let him in.

As we explained in the previous section, ROS Modules were developed in order to lower the programming complexity of each task, removing the need to handle the ROS topics and services, since this will all be done within the modules.

The state machine for the task can be seen in Figure 5.1.

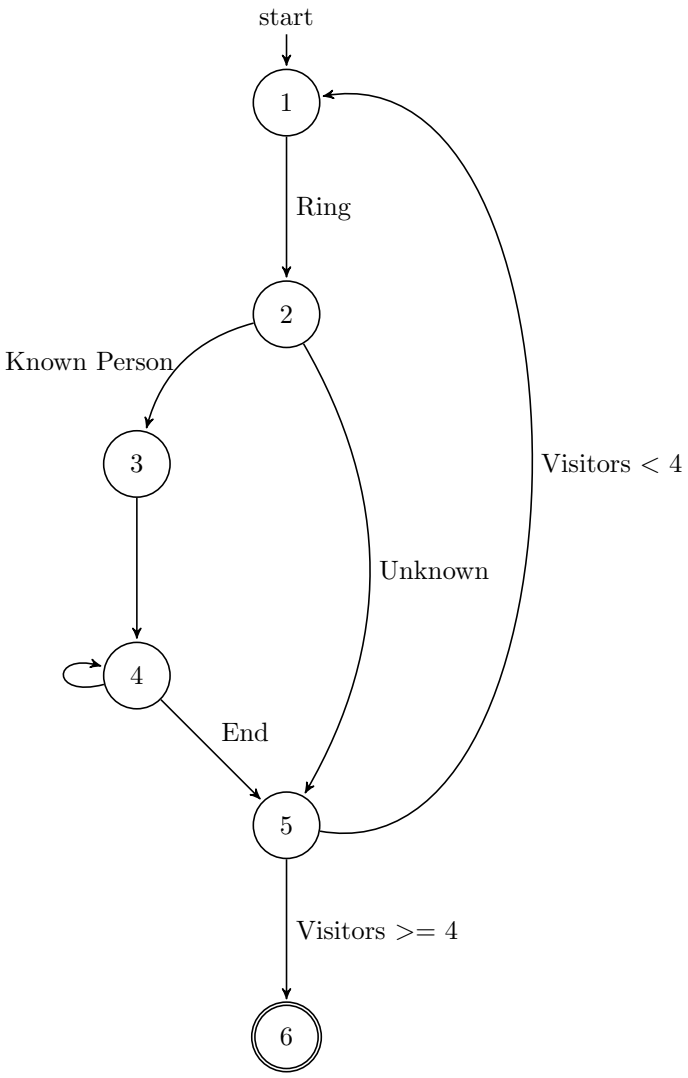


Figure 5.1: State machine for the task. State 4 is expanded in Figure 5.2

The next step is to analyze each state and explain what was done in each of them. The structure of the program is that there is a main thread that is running constantly, and there is a private variable which stores the state. At each iteration, there is a switch case for the state variable, and the actions for the current state are executed. We will now explain what is done in each state.

State 1. Wait Bell In this state, the robot continuously checked if the bell had been rang, by being connected to the house's devices. Whenever the bell was rang, the robot would change its state and proceed to recognize the visitor. The function used in this state is one which belongs to the devices module, and is called `listen_bell()`. After this, the robot asks the visitor to look at the camera, by using the text to speech module.

State 2. Recognize In this state, the person recognition functions are called, applying the algorithms explained in Section 3. The function called, which belongs to the person recognition module, is named `classify_current_person`. Once it finishes, its output is the identified person, who will either be known (Deli Man, Postman or Dr. Kimble), in which case the corresponding action is to greet him, or Unknown, in which case we will dismiss the visitor. In this state, it also checks more than one time if the person classified is correct, until a number of maximum retries which is previously configured.

State 3. Greet In this state, the robot must announce to the visitor that he knows his or her identity and that he will proceed to allow him or her to enter the house. To do this, the text to speech module is used, specifically the function `say()`, changing the welcoming sentence for each of the known visitors. Then, the robot moves closer to the door, using the navigation module, and asks the visitor to open the door.

State 4. Do action For this very general state, a new state machine was introduced, in order to do the different actions required for each visitor. This state machine is the one shown in Figure 5.2. In each of the states, the robot must do different actions. In this section, it is very important for the robot to communicate what he expects of the visitor, although the task is correctly specified and each person knows its role, it is important that the robot states his requests in order to have a much more realistic interactive experience.

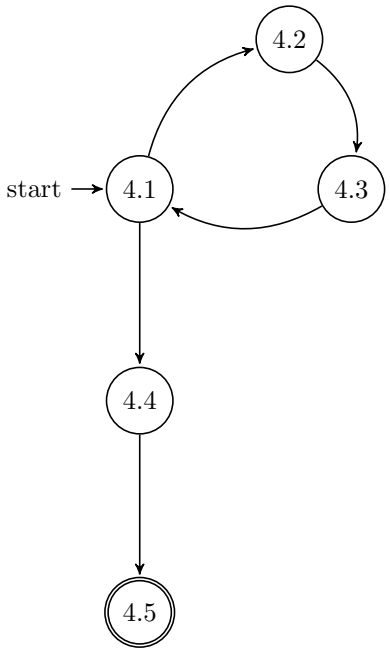


Figure 5.2: State machine for the action state

State 4. Action 1. Request to follow Firstly, the robot will communicate via the text to speech module to the visitor that he requests him to follow him to the corresponding room. This is done using the text to speech function `say`.

State 4. Action 2. Navigate to room The next step is to use the navigation module to go to the room. The robot will avoid the obstacles and move through the shortest path in order to arrive to its destination. Its goal will be a Point of Interest (POI) inside the room which allows the visitor to enter it and have access to whatever it needs to do. The function used belongs to the navigation module, and the function is called `go_to_poi()`.

State 4. Action 3. Wait in room Once the robot will reach the room, it will announce what it expects the visitor to do, for example leaving the breakfast on the kitchen table, delivering the mail or attending the Granny. Then, it will wait for some time until the visitor has finished the action, before it proceeds to the next state. This state is different for each visitor, so a switch case is done and different sentences are stated depending on which visitor is in the apartment.

State 4. Action 4. Navigate to door Once the visitor has finished doing his job, the robot will lead the way to the door. The function used is the same as before, but changing the goal point to the front door.

State 4. Action 5. Finish action This state simply checks that the navigation to the door is finished, and returns to the main state machine.

State 5. Dismiss The robot says goodbye to the visitor, and goes back to its initial waiting state. In this case, it checks whether the number of visitors he has attended is less than 4, in which case it starts over waiting for the bell. Otherwise, the task has concluded and the robot changes to the End state.

State 6. End When the four persons have rung the bell, the robot moves to the end position, indicating that he has finished the task.

5.2 Implementing the state machine

The implementation of this state machine was done inside a loop, which at every iteration checked which state it was in, by using a switch case. An implementation of this switch case is shown in Listing 5.2. The variable which stored the main state was `t2_m_s`. In this code there is also a start task state, which waits for the referee to initialize, which is necessary for the competition. In each state, there is the conditional which changes the state, and a call to the action done in that state. In this implementation, the logging is also shown, since it was necessary to record different information. The action algorithm, which is called when the state is `T2_ACT`, has a similar structure to this piece of code, but using the states defined as the action states.

Listing 5.1: Implementation of the state machine

```
switch (this->t2_m_s){
  case T2_START:
    if(this->referee.execute()){
      this->startTask = false;
     this->t2_m_s=T2_WAIT;
     this->log_module.start_data_logging();
    }
    else
      this->t2_m_s=T2_START;
    break;
  case T2_WAIT:
    if (devices_module.listen_bell()){
          this->t2_m_s = T2_ASKLOOK;
          this->hasCalled = false;
          this->log_module.log_command("ring_bell");
          this->log_module.
              start_logging_images_front_door();
    } else {
          this-> t2_m_s = T2_WAIT;
```

```
    }
    break;
  case T2_ASKLOOK:
      if (this->action_say_sentence("Please look at the
          camera")){
             this->t2_m_s = T2_CLASSIFY;
      }
      else this->t2_m_s = T2_ASKLOOK;
      break;
  case T2_CLASSIFY:
    result = this->classifier_module.
       classify_current_person(label,acc,error_msg);
    if (result) {

      if (error_msg == "" || this->
          classification_retries > this->config_.
          max_retries){
      if (labelToPerson(label)){
        chooseIfCorrectPerson (label,acc);
        if (this->t2_m_s == T2_ACT){
          this->log_module.log_visitor(currentPersonStr
              ());
          this->log_module.
              stop_logging_images_front_door();
          this->classification_retries = 0;
        }
      }
      }
      else {
        this->t2_m_s = T2_ASKLOOK;
        this->classification_retries++;
      }

    }
    break;
  case T2_ACT:
    if (this->action_algorithm()){
      this->t2_m_s = T2_RETURNIDLE;
    }
    break;
  case T2_RETURNIDLE:
    if (this->action_gotoIDLE()){
      this->t2_m_s = T2_FINISH;
    }
    break;
```

```
    case T2_FINISH:
        this->visitors_counter ++;
        if (this->visitors_counter >= this->visitors_num)
            {
          this->t2_m_s = T2_END;
        }
        else {
          this->t2_m_s = T2_WAIT;
        }
      break;
    case T2_END:
      this->log_module.stop_data_logging();
      this->referee.execution_done();
      break;
  }
```

Chapter 6

Tests and Results

In this chapter, several experiments have been made in order to justify the different decisions taken in the design of the person recognition algorithms. These experiments are divided into four sections. Firstly, an explanation of the different face detection methods that were approached, in order to select the face region from the rest of the image, and work from there. Secondly, the face recognition methods that were used in order to recognize the face of Dr. Kimble, which is the one that the robot had to correctly identify. Thirdly, analyze the general person recognition algorithms in the competition, that is, evaluate the capability to distinguish between the four possible visitors in the tasks. Finally, the results of the competitions in Barcelona and Edinburgh are shown.

6.1 Face detection

The first step for recognition is to identify the location of the person within the image. Once we have its location, we can identify the region of the face, to perform a face recognition, or focus on the upper body in order to detect the shirt color. The two methods analyzed are the Histogram of Oriented Gradients (HOG) and the Haarcascade.

6.1.1 Method 1 - Histogram of Oriented Gradients

As explained in Section 3.1, this method is based on the work of Dalal and Triggs, who created a person recognition method by using HOG. The OpenCV libraries have an implementation of this algorithm, which includes a trained classifier for person detection. This function attempts to find all the people in an image and returns a set of rectangles which define the endpoints of the people's locations.
By using this, we obtained some good results, in which the people found matched the person who was in front of the camera, and only one person was detected, as seen in Figures 6.1a and 6.1c. However, in many other cases, more than

(a) Good detection (b) Bad detection

(c) Good detection (d) Bad detection

Figure 6.1: Results of HOG for person detection

one person was found, which was a problem for us since we were not able to know which of the detected regions was the correct one. This problem is seen in Figures 6.1b and 6.1d, where due to the position of the arms, more than one person is detected, meaning that we would not be able to find the exact location of the face.

6.1.2 Method 2 - Haarcascades

As explained in Section 3.1, this method is based in the algorithm introduced by Viola and Jones in 2001, which uses a cascade classifier to detect faces in images. As well as with the previous method, OpenCV provides an implementation of this algorithm, in which a pre-trained cascade classifier must be loaded from an XML file, and then used to identify the faces present in the image. When calling this function, it returns a set of rectangles with all the endpoints of the faces found. In this case, we always detected at most one face, which solved the previous problem of multiple detections.
As it is shown in Figures 6.2a and 6.2c, the results were quite satisfying, despite some frames in which the face was not found, like in Figure 6.2b. We also

(a) Good detection (b) Bad detection

(c) Good detection (d) Bad detection

Figure 6.2: Results of Viola-Jones algorithm for person detection

attempted to check whether a non-frontal face would be correctly detected, but as it can be seen in Figure 6.2d, the person must be looking at the camera in order for it to work properly.

6.2 Face recognition

Having the face detected, we will now explore two different face recognition methods in order to correctly classify whether the observed face is of Dr. Kimble or if the face is an unknown person's.

6.2.1 Method 1 - Dlib

When we attempted this method, we observed that the results were not as good as we expected. We had a set of images used for training, and then we attempted to test the classification. When we tested this methodology while in the laboratory, we used a usb camera, with which we took pictures from different people. The outcome was that the classification rate was very low, and we had expected higher results.

We observed that the images that had previously been seen during the training were correctly classified, but new images were constantly misclassified.

When attempting to use this method in the competition in Barcelona, due to the low resolution of the front door camera and the distance from it to the visitor, the correct classification rate was even lower and almost always misclassified Dr. Kimble. Out of 66 images that we had collected, none were correctly classified.

6.2.2 Method 2 - face_recognition package

As mentioned previously, in this method we do not need to use our face detection because the package already includes this functionality, so we can work directly with the image obtained from the camera and call the face detection function that the package provides.

When testing this method, we observed that the results were much better: Out of 66 images of Dr. Kimble that we had collected, 63 were correctly classified, which means a success rate of 95.5%.

The downside of this method is that the time to calculate this was slightly higher. We observed that the time elapsed between sending the image and obtaining a result of the classification was of 0.75 seconds, while in the previous methodology we obtained a time of around 0.1 seconds.

6.3 Person recognition

The camera used in the competition was different than the one used for testing the face recognition methods, and the distance from the person to the camera also changed. This meant that some of the results changed before and after the competition. Every time we wanted to classify a new visitor, we followed the algorithm described in Chapter 3.2, so four steps needed to be done. First, find the person's face, second, find the body, third, find the shirt's color, and finally, attempt to recognize the face. In Figure 6.3a we can see an example of the face being found, the shirt being found and the color correctly detected. By knowing the shirt's color and recognizing the person's face, we could identify which of the visitors had arrived.

Most of the results in this case were satisfactory, but we discovered that when a person wore a cap which partially covered his or her face, our algorithm could not find the face in the image, which is what happens in Figure 6.3d. This is why we added a voice line to the robot's code in which we asked the person to look directly to the camera, in order to solve this problem.

Another problem that we discovered in this section was that if the unknown person was wearing a yellow or a white shirt, he would be classified as either the Postman or the Deliman, and allowed to enter. This was due to the fact that our algorithm only took into account the shirt color, and not the explicit uniform, which contains the logo of the company.

(a) Good detection (b) Good detection

(c) Good detection (d) Bad detection

Figure 6.3: Results of face, body and color detection during the ERL competition

6.4 Task execution

The task specification states a list of achievements and penalized behaviors. During the two competition, the goal was to obtain the maximum achievements and the least penalizations. The list, extracted from the ERL competition rulebook, is the following:

The set A of *Achievements* are:

- The robot opens the door when the door bell is rung by Dr. Kimble and correctly identifies him.
- The robot opens the door when the door bell is rung by the Deli Man and correctly identifies him.
- The robot opens the door when the door bell is rung by the Postman and correctly identifies him.
- The robot opens the door when the door bell is rung by an unknown person and correctly identifies the person as such.

- The robot exhibits the expected behavior for interacting with Dr. Kimble
- The robot exhibits the expected behavior for interacting with the Deli Man.
- The robot exhibits the expected behavior for interacting with the Postman
- The robot exhibits the expected behavior for interacting with an unknown person.

The set P of *Penalized Behaviours* are :

- The robot fails in making the visitor respect the proper rights
- The robot generates false alarms.
- The robot fails in maintaining the original state of the environment.
- The robot requires extra repetitions of speech.
- The robot bumps into the furniture.
- The robot stops working.

As it can be seen, the total number of achievements obtainable are 8, and the total number of penalties are 5. The goal was to maximize achievements and minimize penalties. The task was be repeated a total of 5 times, and the final score was calculated as the median of all the scores.

6.4.1 November 2017 - Barcelona

During the competition of Barcelona, there was some setup time in which we asked the person who would act like Dr. Kimble to walk in front of the camera and we recorded some of his images, in order to have a database. Each task was supposed to be run only four times, and there was some time at the end to repeat one of the runs if desired. In our case, we repeated this specific task so that our scoring would improve, so a total of 5 runs were done.

In the first competition, our face recognition library was not able to identify Dr. Kimble, this meant that it would not open the door for him, nor exhibit the expected behavior. The first two runs, however, showed a worse result than expected, because since we were trying to adjust the person recognition library, one of the runs assumed that all of the people were Dr. Kimble. After seeing that with the new camera, with its resolution and distance from the visitor, our algorithm was not able to perform a good person recognition, we disabled the functionality and forced that the person's face would always be unknown. This allowed us to achieve a result of 6 points out of 8 on the last three executions. In the first run, we obtained two points for recognizing the Postman and Dr. Kimble, and one extra point for doing the expected action when the Postman entered. We did not do the correct actions for when Dr. Kimble entered because

Visitor	RUN 1	RUN 2	RUNS 3,4,5
Postman	Postman	Postman	Postman
Deli man	Dr. Kimble	Dr. Kimble	Deli man
Dr. Kimble	Dr. Kimble	Deli man	Unknown
Unknown	Dr. Kimble	Dr. Kimble	Unknown

Table 6.1: Predicted visitor output in the Barcelona competition

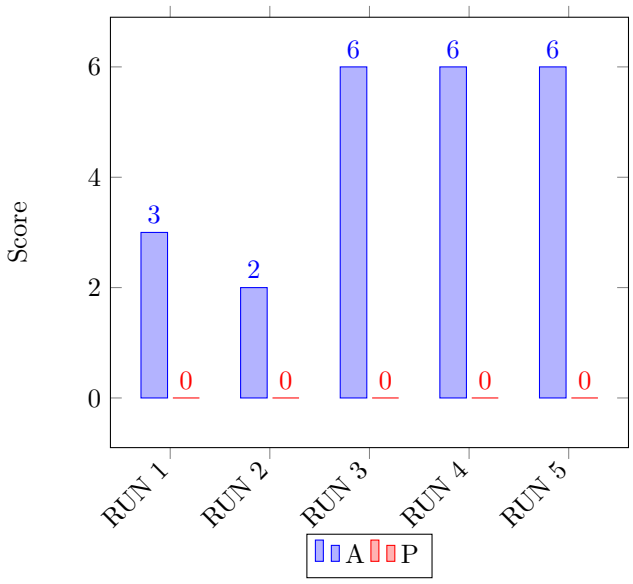


Figure 6.4: Results of the Barcelona competition

we did not have time to implement a module that detected when a person left a room. In the second run, we only recognized correctly the Postman, and this is why we only obtained two points. In the last three runs, We recognized three out of the four people correctly, and did all the actions accordingly.

The final result, which was calculated as the median, was of 6/8 points.

6.4.2 Upgrades

- **Change method from Dlib to face_recognition** Due to the fact that face recognition did not provide good results in the Barcelona competition, we explored a new method which proved to be more successful. We modified the library implementation, by using the face_recognition function in Python, and then wrapping the code with C++. The rest of the library implementation remained the same, and the only modification was the call to the face recognition function.

- **Use module for detecting if someone leaves the room** The expected behavior for Dr. Kimble includes detecting when he leaves the room and then following him to the door, whereas the rest of the visitors must follow the robot to the door. In order to do this, a new module was developed by another member of the team that used an image subtraction technique to detect the moment when the doctor left the room. This module was included in the task state machine in order to achieve this behavior.

6.4.3 January 2018 - Edinburgh

At the end of January, with the upgrades implemented on the robot, there was the competition in Edinburgh. Using the developed code, we did 5 executions of the task, the results are the ones shown in Figure 6.5. All the attempts had a full result, except the fourth one, in which the person who acted as Dr. Kimble was wearing a white T-Shirt. Our algorithm failed at recognizing him and so the robot thought it was the Deli Man who knocked on the door.

As well as in the competition in Barcelona, the final result was calculated as the median, which in this case, provided a result of 8/8 points.

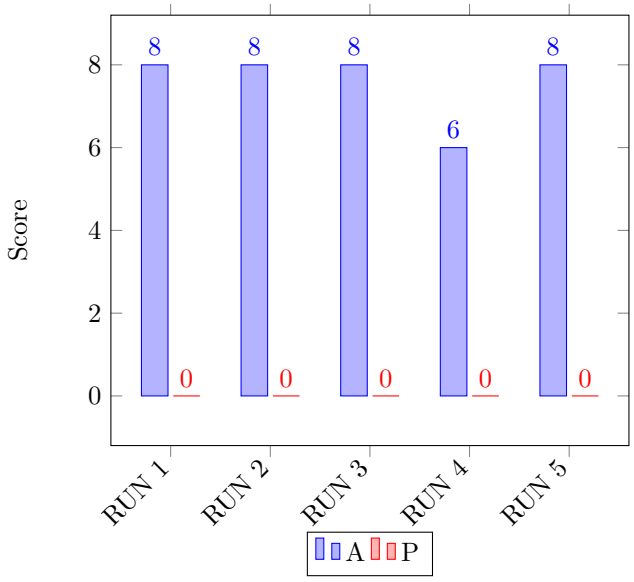


Figure 6.5: Results of the Edinburgh competition

6.5 Accuracy tests

Some weeks after the competitions, different tests were made in order to evaluate the accuracy with the different methods. For each of the experiments

Face detection results					
Algorithm	Mean execution time	Detected	Undetected	Multiple detections	Accuracy
HOG Algorithm	0.341617 s	11	35	4	**22%**
Viola Jones Algorithm	0.132154 s	47	3	0	**94%**

Figure 6.6: Face detection results

Face recognition results					
Algorithm	Mean execution time	Kimble	Unknown	Undetected	Accuracy
dlib library	0.380889 s	6	44	0	**12%**
face_recognition library	0.758071 s	48	0	2	**96%**

Figure 6.7: Face recognition results

we wanted to test, we stored 50 frames of the frontal face of the person to detect, and then stored the results of each of the methods.

As we can see in Table 6.6, the Viola Jones algorithm yields an accuracy of 94%, while the HOG algorithm only reaches a 22% accuracy. Furthermore, the Viola Jones algorithm is faster than the other one.

For face recognition, as shown in Table 6.7, the dlib library provides only an accuracy of 12%, while the python face_recognition library reaches un accuracy of 96%. However, the execution time is much slower for the latter.

Chapter 7

Conclusion

7.1 Discussion

In this section we will review if the objectives defined in the project have been successfully achieved. Let's recall that our main objective was to *implement person recognition software on a service robot by using several tools and modules so that it can answer the door for different visitors.* This objective was successfully achieved and now we will show what the previously defined indicators say about the results.

- **Implement a face detection algorithm, with an accuracy of 90%**. By using the Viola-Jones face detection method we have obtained good results when the face was looking forward, with an accuracy of 94%. In the case of the Deli Man, it was observed that the person not only had to have a frontal position, but also needed to look directly at the camera, due to the fact that the uniform hat prevented the algorithm to correctly detect the facial features.

- **Implement a person recognition algorithm, with an accuracy of 90%**.By using the Python face recognition package, our person recognition accuracy increased greatly, and in the Edinburgh competition this result was proven. As mentioned in Chapter 6, our final accuracy was of 96%. We understood that it was necessary to invest more time in the classification step in order to obtain better results, and due to the fact that the response time was still less than 1 second, we are satisfied with the final result concerning face recognition.

- **Implement a state machine with ROS, so that the robot reacts accordingly to the visitors**. In the second competition, the robot successfully did all the actions that were expected of him for each of the visitors. The robot communicated correctly, announced the visitor when necessary and requested to be followed in order to guide the visitor through the flat. The only drawback of this objective was that the behavior for

waiting for Dr. Kimble to finish attending Granny Annie and then following him to the front door was only implemented after the first competition, which is explained in the previous chapter.

- **Use the developed software in the competitions and achieve the maximum score**. In the first competition in Barcelona, although the score was not as high as we would have wished, the median of all the runs provided us with a score of 6/8 points. This was also the maximum score achieved by other teams in the local round, so we were the partial winners of the task. In the second competition in Edinburgh, we obtained the maximum score of 8/8, resulting in winning this specific task of the ERL competition. The prizes were awarded during the European Robotics Forum in Tampere, Finland, 13-15 March 2018.

7.2 Future work

We will now review which improvements can be made, and what the next steps for this project will be.

- **Find logo of the uniform.** As seen in Chapter 6, some of the mistakes in the competition were due to the fact that the visitor was wearing a t-shirt of the color of the uniform. This could cause many security threads since we are only checking for the shirt color to match the uniform, and not the logo. In a controlled environment like the competition, it was only necessary to match the color, but if the software were to be deployed in a real home, it would be necessary to improve it by detecting the logo of the uniforms. This part could be done in many different ways, from the moment where the shirt position is detected, the pixels in the logo could be found, to distinguish if the colors are the same. For example, as seen in Figure 7.1a, we could detect a set of pink pixels, or in Figure 7.1b, we could attempt to detect a gray rectangle. Another possible, and more reliable method would be to attempt to parse the letters in the logo, by detecting the word "ROCKIN", we could be positive that the logo is correct. The problems with this approach is that the resolution of the image is quite low, so it is possible that the letters cannot be seen correctly, and also that the shirt is a deformable objects, so wrinkles and folds my alter the words that the camera perceives.

- **Improve body detection** As seen in Chapter 6, one of the problems was the inability to detect the face of the visitor when the person was not looking directly at the camera, either by being in a sideways position or due to the fact that he or she was wearing a hat that covered the facial features. During the competition, the workaround to this issue was to ask the visitor to look directly to the camera, which was something that was allowed and improved the communication between the robot and the visitor. However, one way to improve this would be to detect the body of

(a) Logo in the shirt of the Deli Man (b) Logo in the shirt of the Postman

Figure 7.1: Different logos of the visitors

the person directly, instead of having to detect the face as well. In order to do this, the same method of face cascades described previously can be used, but using a different cascade file specifically trained to detect bodies. This approach would increase our detection rate for when the person is not looking at the camera in order to have a much more robust person recognition algorithm.

- **Standardize error handling** This project was developed simultaneously as other pieces of software, all of which had to run on the TIAGo robot and which could some times contain mistakes, such as inability to communicate with other ROS nodes, failure when initializing the libraries or task-specific errors. There is no pre-defined way of error handling, so each of us attempted a different approach, such as printing the information through the standard output channel, returning an error code or blocking the program until the problem was resolved. However, it would be much better to have a standard way of doing this, implemented in all the pieces of software used in the competition so that error handling is much easier to resolve.

After finishing the competitions, IRI bought the TIAGo robot so that more research in robotics could be made with it. It is currently being used for a socially assistive robotics project, in which the robot plays a cognitive game with a human player. The game is designed for Alzheimer and dementia patients, for whom it has been proven that playing games improves the cognitive capabilities. In the game, TIAGo guides the patient through possible solutions, by providing speech and gesture support, giving hints or telling the human which mistakes were made. This feedback is given depending on the player's reaction time and his results.

It was recently announced that the ERL competition will continue for season 2018-2019, and the IRI team signed up for participating in the local round that will take place in October 2018 in Madrid, in the context of the International Conference on Intelligent Robots and Systems(IROS). It will be a great oppor-

tunity to include these improvements in the new competition and have a much more robust algorithm in order to obtain the maximum score again.

7.3 Conclusion

During the development of this project we have seen that delegating the task of welcoming visitors to a robot is possible in a controlled environment. Facial detection and recognition and clothes color identification are three robust methods that may allow robots to replace humans in tasks such as this one. Nevertheless, it is still not possible to have this task done correctly in a real world environment, due to the security issues that rise from it. We are still skeptical of trusting a robot to decide when to open the door, knowing that there might be some chance of error. One possible solution to this concern is to have the robot interact with visitors, but have humans in charge of the decision of letting in some people or others.

We have observed that the field of service robotics is growing and many tasks can be replaced by robots. Specifically, in a household environment, robots can provide assistance for older people who are unable of being independent. This will ease the life of the elderly, and help them be more autonomous. However, we must keep in mind that although they might provide support and assistance they can never substitute human company, which is one of the main concerns of this advance in socially assistive robotics.

The results of the competitions were very satisfactory although the executions were not perfect: in both competitions we obtained the maximum score in comparison to the other teams, allowing the IRI team to win the prize for the second task of the ERL.

Appendix A

Sustainability analysis

A.1 Self assessment of the current domain of sustainability competition

The sustainability report of any project must include three basic sections, which analyze the environmental, economic and social impact. Technology grows at a rapid pace, rendering products obsolete in very little time, which means that when developing technology, we must consider whether the platform used to develop our product will still be useful some years from now, and when buying technology, we must think about the consequences of renewing our gadgets every few years. Another aspect to take into account is the environmental impact of technology : on one side, the extraction of materials to develop hardware are not always ethical, since they come from third world countries in unstable political situations, and on the other hand, planned obsolescence is causing to waste technological products and generating landfills of e-waste all around the world. The social aspect of each project must also be taken into account, due to the fact that, as we are starting to see, many jobs can be substituted with technological solutions, from self-checkout machines at supermarkets to automatic question and answering software in websites.

I think one of my strengths is that I am very concerned about the environment and I try not to buy new devices if it is not strictly necessary. However, I yet have to bring these personal beliefs to practice when I am the developer of technology products. I am also very concerned with the job destruction that technology may cause in the near future, mainly because I want to see the change of paradigm that will happen, and in my opinion, it will create a much fairer world, by reducing the number of working hours a week. I don't see as a negative thing to replace people with technology in the workplace, because although I know that in the short term, this will result in higher unemployment rates, it will later be a benefit for all society, maybe even introducing the minimum wage idea for every citizen. However, in order to be fully consistent with my beliefs, I have to learn to provide a possible transition for the people whose job becomes

obsolete due to the new technology developed. Finally, one of my weaknesses is that I am not familiar with planning an economic budget and assessing the cost of product development, due to my background, but I understand that it is a necessary aspect in order to fully understand the sustainability impact of a project.

A.2 Economic Dimension

A.2.1 Budget

For the most part of the development of the project, a Macbook Pro 15' was used. In it, the libraries were implemented and developed, as well as the ROS modules necessary to make the robot function. In order to use the TIAGo robot, PAL Robotics offered a loan to some of the participating teams, for the price of 650€ per month. The robot was loaned from October to January. The development time of the project was around 575 hours, with which the human resources and laptop consumption parts of the budget were estimated. We added a 15% of contingency, in order to have some extra budget if any unexpected problems would arise. Finally, regarding some possible unforeseen events, we took into account that it might be possible to add some hours for debugging, for which the human resources must be paid, the possibility of breaking the computer and having to buy a new one. The TIAGo robot had an insurance so that if we broke it we did not have to pay the full cost. However, in order to be careful and not allow this to happen, both the computer and the robot were always managed with the utmost care, in order not to damage them. Concerning the extra debugging hours, the plan was to deviate as little as possible from the schedule.

The hardware costs were calculated in the following way; taking into account that the hardware will be amortized in 4 years, and that each year there are approximately 250 workable days, at 8 hours each day, that is a total of 8,000 hours. However, since the TIAGo robot was loaned, we will only take into account the workable hours of the 4 month period of October to January, which makes a total of 85 working days, which are around 700 hours.

The two hardware components used, and their price per hour are the following.

The software used was all open source, meaning they have no cost.

Hardware	Total price	Price per hour
Macbook Pro 15'	2,600.00 €	0.325 €/h
Loaning TIAGo robot (4 months)	2,600.00 €	3.714 €/h
TIAGo robot insurance (4 months)	1,200.00 €	1.714 €/h
TIAGo robot total cost (4 months)	3,800.00 €	5.428 €/h

Details	Hours	Price per hour	Total
DIRECT COSTS			
Literature review and documentation			
Macbook Pro 15'	82 h	0.325	26.65
Library design			
Macbook Pro 15'	63 h	0.325	20.50
Library implementation			
Macbook Pro 15'	98 h	0.325	31.85
ROS modules design and implementation			
Macbook Pro 15'	56 h	0.325	18.20
Task state machine implementation			
Macbook Pro 15'	32 h	0.325	10.40
Setup for competition			
TIAGo robot	16 h	5.428	86.85
Macbook Pro 15'	16 h	0.325	5.20
Testing			
TIAGo robot	23 h	5.428	124.84
Macbook Pro 15'	23 h	0.325	7.48
Competition			
TIAGo robot	31 h	5.428	168.27
Macbook Pro 15'	31 h	0.325	10.08
Improvements			
Macbook Pro 15'	49 h	0.325	15.93
Outlining project document			
No cost	15 h		
Gathering accuracy data in different test sets			
Macbook Pro 15'	53 h	0.325	17.23
Writing the document			
No cost	57 h		
Total Human resources			
Intern at laboratory	575 h	6.00/hour	3,450.00
INDIRECT COSTS			
Laptop consumption	575 h	90 W * 0.11 /kWh	5.69
SUB-TOTAL DC+IC			3,999.17
CONTINGENCY	15%		599.88
UNFORESEEN EVENTS	Probability	Cost	Total
100 Extra-hours needed for debugging	30%	600.00	180.00
Broken computer	5%	2,600.00	130.00
TOTAL COST			4,909.05 €

A.2.2 Reflection

The total cost of this project is high, since it is developed in a research laboratory, and there is enough money to cover the total cost for this project. However, a project

as engaging as this, which will allow all the team members to participate in an european robotics competition and get to understand better the world of robotics, is worth the economic cost shown in the budget analysis.

The current state of the art of vision algorithms usually develops systems using GPU with a high processing power. In our case, these GPUs were not used for the final version of the software. Another positive economic aspect is that the version of the libraries used are all the most recent, which means that they won't become obsolete in some time, and when they do, all the code is documented so that it is easily updatable in case the libraries' API change.

All projects in the robotics field are expensive, due to the inherent price of buying or renting a robot. However, for this competition, PAL Robotics provided loans of the TIAGo robot for the teams that participated in the competition, at cheaper rates than usual. This allowed to develop the product without spending as much as other research groups, and also the possibility of working with a TIAGo robot so as to decide whether further research with it would be of interest. This, however, implied the risk of damaging the robot, in which case it would have to be bought, but we were very conscious about that and took all necessary precautions such that the robot would not be broken.

This project was fully paid for by IRI, because since it is a research laboratory, it is interested in providing help for developing robotics projects.

The current state of the art of person recognition software is usually tested on a fixed computer, so there is no problem in having multiple GPUs on the system in order to both train and run the neural nets. However, in our case, the software had to run on a laptop which was on top of the TIAGo robot, so the system was limited by size. Our solution solves this by using the small Macbook computer, and it is able to make the robot autonomously recognize the different people without needing a bigger hardware setup.

A.3 Environmental Dimension

As mentioned in the previous section, the TIAGo robot was loaned from PAL Robotics, because buying one only for the competition wouldn't be environmentally conscious. All the equipment bought specifically for the competition will be later reused for some years and future projects, kept in the IRI research laboratory.

In some years, if the hardware is no longer useful, it will be properly recycled in a green point, instead of the general trash, in order not to generate e-waste.

The energy used during the development of the project was exactly that which was needed, not wasting any extra energy and thinking about the environment.

The implementation of this project doesn't directly show any way in which the environmental impact will be improved. However, the thought of having service robots might improve the energy waste that happens daily in every home. For example, a robot could detect if the lights are on when nobody is home, or when the energy consumption is higher than the required.

A.4 Social Dimension

Regarding the personal impact of this project, it will greatly help me understand more about the world of robotics and computer vision, which are the fields in which I want to work in the future. Participating in a European robotics competition will also help me develop very important soft skills such as teamwork and effective communication. Finally, this project will help me enter the world of software engineering, as it is the first large scale project I've ever developed.

Concerning the general impact of the project, we can consider that there is a need for developing efficient recognition software in service robots so that many household tasks can be simplified or delegated. Providing a TIAGo robot with this ability will make it much more human-like and thus prepared to interact with any person, regardless of the age or knowledge.

Concerning the impact on other people, we must first take into account the fact that the software will not be used after the competition. This is due to the fact that TIAGo robots are expensive (as seen in Section A.2) and for the moment being, they are of much more use in research labs than in personal homes. However, the developed code is perfectly functional and it could be adapted in order to use it in someone's home.

Having a service robot at home with the ability to recognize visitors will allow people to have less things to worry about and for example eliminate the need to go to the door each time someone wants to enter. This will be helpful for people with reduced mobility or elderly people. However, constantly storing the images from a camera might cause a reduction in privacy and might need to be regulated in order to ensure that nobody can access this data.

Appendix B

Tiago datasheet

TIAGo©

TECHNICAL SPECIFICATIONS

Simulation model available at:
wiki.ros.org/Robots/TIAGo

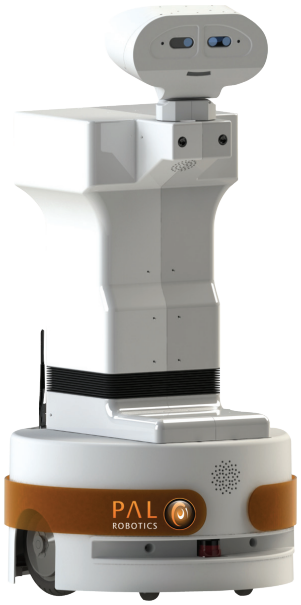
GENERAL FEATURES

Height	110 - 145 cm
Weight	70 Kg
Footprint	ø 54 cm

DEGREES OF FREEDOM (DoF)

Torso lift	1	Arm	7
Mobile base	2	TOTAL (without end-effector)	12
Head	2		

IRON



STEEL

TITANIUM

CONFIGURATION	IRON	STEEL	TITANIUM
Mobile Base	✓	✓	✓
Navigation Laser	5.50m	5.50m	10m
Lifting Torso	✓	✓	✓
Pan-tilt head	✓	✓	✓
7 DoF arm		✓	✓
End-effector		Parallel Gripper	5 finger hand
Force/torque sensor			✓

* Upgrade Kits available in order to evolve the Iron version to Steel and Titanium

PAL ROBOTICS

C/Pujades 77-79, 4º 4ª
08005 Barcelona, Spain

Tel: +34 934 145 347
Fax: +34 932 091 109

info@pal-robotics.com
www.pal-robotics.com

TIAGo©

TECHNICAL SPECIFICATIONS

Simulation model available at:
wiki.ros.org/Robots/TIAGo

BODY	Arm payload (at full extension)	2 Kg
	Arm reach (without end-effector)	87 cm
	Torso lift	35 cm
MOBILE BASE	Differential drive	✓
	Max speed	1 m/s
	Operation enviroment	Indoor
CONNECTIVITY	Wireless connectivity	802.11 n/ac 2x2 Dual Band Wi-Fi Bluetooth 4.0
ELECTRICAL FEATURES	Battery 36V 20Ah	1 battery / 2 batteries
	Battery Autonomy	4-5h / 6 - 10h
SENSORS	Base	Laser 5.5m or 10m range, rear sonars 3x1m range
	IMU (Base)	6 DoF
	Motors	Actuators current feedback
	Torso	Stereo microphones
	Head	RGB-D camera
COMPUTER	CPU	Intel i5 / i7 Haswell
	RAM	4 GB / 8 GB / 16 GB
	Hard Drive	60 GB / 120 GB SSD
SOFTWARE	OS	Ubuntu Linux LTS
	Open source middleware	ROS
	Periodic updates/patches	✓
	Arm with	position / velocity / effort control
	Eye-hand calibration suit	✓
SUPPORT	Training	On demand
	Maintenance plan	On demand
	Ticketing system	✓
	Online helpdesk	✓
	Research community forum	✓
EXTENSIBILITY	Laptop tray	✓
	Mounting Points	On head and laptop tray
	USB ports	1x USB3, 1x USB2
	Ethernet ports	2x Gigabit
	Power supply	12V / 5A
OPTIONALS	End-effector	5 finger underactuated hand / Parallel gripper
	Wrist sensor	Force/torque sensor
	Whole body control	✓

PAL ROBOTICS

C/Pujades 77-79, 4º 4ª
08005 Barcelona, Spain

Tel: +34 934 145 347
Fax: +34 932 091 109

info@pal-robotics.com
www.pal-robotics.com

Bibliography

[1] Omaima Al-Allaf. Review of face detection systems based artificial neural networks algorithms. In *The International journal of Multimedia and Its Applications*, volume 6, 03 2014.

[2] David Arthur and Sergei Vassilvitskii. K-means++: the advantages of careful seeding. In *In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[3] Meysam Basiri, Pedro Lima, Matteo Matteucci, and Lucha Iocchi. *European Robotics League for Service robots*. 2016.

[4] Peter Belhumeur, Joao Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 43–58, 01 2006.

[5] Miyoung Cho and Youngsook Jeong. Face recognition performance comparison between fake faces and live faces. In *Soft Computing*, volume 21, 01 2016.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision Pattern Recognition*, volume 1, pages 886–893, 07 2005.

[7] Mady DELVAUX. Draft report with recommendations to the commission on civil law rules on robotics. Technical report, European Parliament, 05 2016.

[8] euRobotics. European robotics league website. `https://www.eu-robotics.net/robotics_league/about/index.html`.

[9] Open Source Robotics Foundation. Ros. `http://www.ros.org/`.

[10] Adam Geitgey. Face recognition. `https://pypi.org/project/face_recognition/`, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 7, 12 2015.

[12] Ming hsuan Yang, Dan Roth, and Narendra Ahuja. A snow-based face detector. In *Advances in Neural Information Processing Systems 12*, pages 855–861. MIT Press, 2000.

[13] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[14] Rabia Jafri and Hamid Arabnia. A survey of face recognition techniques. 5:41–68, 06 2009.

[15] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[16] Divyarajsinh Parmar and Brijesh Mehta. Face recognition methods & applications. In *International Journal of Computer Technology and Applications*, volume 4, pages 84–86, 01 2014.

[17] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The feret evaluation methodology for face recognition algorithms.

[18] Ian R. Fasel and Javier Movellan. A comparison of face detection algorithms. pages 1325–1332, 08 2002.

[19] Henry Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, pages 203–208, 01 1996.

[20] Nicholas Roy, Gregory Baltus, Dieter Fox, Francine Gemperle, Jennifer Goetz, Tad Hirsch, Dimitris Margaritis, Mike Montemerlo, Joelle Pineau, Jamie Schulte, and Sebastian Thrun. Towards personal service robots for the elderly. 10 2000.

[21] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 09 2014.

[22] Matthew Turk and Alex Pentland. Eigenfaces for recognition. In *Journal of cognitive neuroscience*, volume 3, pages 71–86, 01 1991.

[23] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, 2001.