Final Master Thesis

MASTER'S DEGREE IN AUTOMATIC CONTROL AND ROBOTICS

Real-Time Collaborative Robot Control Using Hand Gestures Recognised By Deep Learning

MEMORY

Author : Leire Amenabar Echave

Supervisor : Prof. Cecilio Angulo

Date : June, 2020

ETSEIB

Escola Tècnica Superior

Enginyeria Industrial de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

BARCELONATECH

elektroautomatik

Abstract

In an ever-changing and demanding world new technologies, which allow more efficient and easier industrial processes, are needed. Furthermore, until now, traditional vision technologies and algorithms have been used in the industrial area. These techniques, even though they achieve good results in simple vision tasks, they are really limited since any change in the processed image affects their performance. For example, in code reading tasks, if the code has a mark or it is not completely visible, the piece with the code would be discarded which leads to losses for the company. These kind of problems can be solved by the implementation of machine learning techniques for vision purposes. Moreover, these techniques learn from example images and even though a perfect performance is difficult to get, machine learning techniques are much more flexible than traditional techniques. Even though the term machine learning was coined for the first time in 1959, until now, these techniques have barely been implemented in the industrial area. They have mostly been used for investigation purposes.

Apart from the new vision techniques, new types of robots are being implemented in industrial environments such as collaborative or social robots. On the one hand, collaborative robots allow the workers to work next to or with the robot without any type of physical interference between them. On the other hand, social robots allow an easier communication between the robot and the user which can be applied in different parts of the industry such as introducing the company to new visitors.

The present project gathers information in regard to the analysis, training and implementation of a vision artificial neuronal network based software called ViDi Cognex software. By the use of this software, three different vision tasks were trained. The most important one is the hand gesture recognition task since the obtained hand gesture controls the action performed by the YuMi robot, which is programmed in RAPID language. It is believed that the development of the different artificial neuronal networks with industrial purposes can show the applicability of machine learning techniques in an industrial environment. Apart from that, the hand gesture recognition shows an easy way to control the movements of a robot which could be used by a person with no knowledge of robots or programming. To finish, the use of a two arm collaborative robot, could show the potential and versatility of collaborative robots for industrial purposes.

Keywords: Artificial neuronal network, ViDi software, YuMi robot, hand gesture recognition, synchronized movements and TCP/IP connection.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

DOF Degrees Of Freedom

GUI Graphical User Interface

IK Inverse Kinematics

OMPL Open Motion Planning Library

POI Point-Of-Interest

R-CNN Regions with Convolutional Neural Network

ROS Robot Operating System

UPC Universitat Politècnica de Catalunya

DMP Dynamic Movement Primitives

IT Information Technology

TCP Transmission Control Protocol

IP Internet Protocol

HRC Human Robot Collaboration

NN Neuronal Network

CNN Convolutional Neuronal Network

ANN Artificial Neuronal Network

OSI Open System Interconnection

UK United Kingdom

US United States

NHG Number Hand Gestures

EA ElektroAutomatik

EAVS EA Vision Studio

PC Personal Computer

TP True Positive

TN True Negative

FP False Positive

FN False Negative

ROI Region Of Interest

UI User Interface

Chapter 1

Introduction

1.1 Motivation

In order to accomplish a specific task, a set of actions are required. The main motivation for this project is to perform an intuitive control of a collaborative robot by the use of common knowledge hand gestures. This way, language barriers, spoken and programming ones, are avoided and an easier communication between the person and the robot is obtained. This project should be implemented and tested by people of different ages and cultures in a technological fair during several days. Hence, an easy understanding and interaction with the robot are a must in this project which should be obtained by the use of the hand gestures for the control of the robot. Nowadays, this degree of performance is reachable by the use of deep learning methods which allows to process complex images.

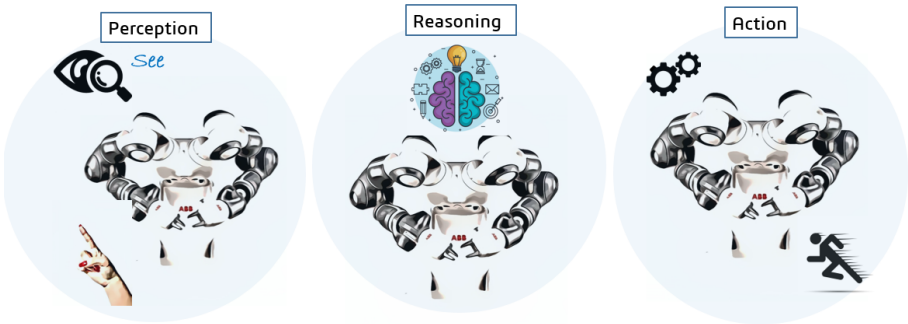


Figure 1.1: Key elements of cognition and process of the robot

To obtain these results, the robot should follow the key steps of cognition (see Figure 1.1). Firstly, the robot is recognising the hand gesture performed by the user. Next, this image is processed and the action to be performed is selected. Finally, the selected action is performed by the robot to complete the whole process of the robot performance.

This project highlights the usability of social robots when it comes to a more interactive, user-friendly and assistive control of robots. The user, without any knowledge of technical skills should be able to

control the robot with intuitive hand gestures. Moreover, even though social and cognitive robots are a trend in robotics, they are barely used in industrial fields and less in production lines. The same situation happens with deep learning methods. Despite vision processing has been used since many years ago in industry and deep learning methods show a better performance than the previous existing algorithms, they have not still been intensively implemented in industry. This project aims to be helpful to show the applicability and good performance of social robots and deep learning methods in an industrial area.

1.2 Objectives

This project aims to reproduce a whole process starting from the recognition of hand gestures (Perception), by the use of deep learning methods, to continue with the selection of the proper action to be performed (Reasoning) and to finish with the accomplishment of the commanded tasks (Action) in real time. The tasks to be performed by the robot will depend on the commanded hand gesture, which will be classified in two groups, either movement result or second action result. Movement result class refers to when the robot is just performing a certain movement, whereas the second action result refers to when the robot has a second reasoning task to be performed. In order to accomplish this general objective, the below listed specific milestones should be defined:

- Evaluate and select best methods required for perception and reasoning.
- Develop a framework that could interpret the user's instruction in a right manner.
- Develop a framework that could perform actions in a dynamic environment.
- Develop a task planning framework that could define its own problem statement.
- Develop the connections between different programs.
- Use RobotStudio to program the tasks to be performed by the robot.
- Develop a framework that runs in real time.

Another goal in this project is to develop an easy and attractive communication with the robot. As far as the robotic platform should be exposed and working in a technological fair, the defined gestures to be controlled by the robot should be easy and natural for the user and the tasks to be carried out by the robot should be interesting and attractive. By accomplishing these objectives people should be attracted to the showroom, so the overall robotic framework and the hand gesture recognition application will be tested by more people.

1.3 Scope

The scope of this project is to develop a robotic, computer vision and deep learning framework which is able to recognise human hand gestures by the use of ViDi software to perform some predefined tasks as well as to test the framework on the YuMi robot from ABB in a technological fair. The project

would have two stages defined by the deep learning part of the framework, the training phase and the implementation phase. This project is developed to include the following aspects:

- Select hand gestures and get data from different participants.
- Define a task to be performed for each hand gesture.
- Recognize hand gestures to select the task.
- Decide the final setup of the framework.
- Develop the program to control the robot reasoning.
- Develop the path planner program.
- Train the artificial neural network (ANN) for hand gesture recognition and tasks.

1.3.1 Used Methodology

In order to engage in the essential research questions of this project, a global research context has to be chosen. As reported by [1], there is generally a research method for each inquiry. However, it can happen that the global research method is constructed by more than one research procedures. The following paragraph presents an analysis of the chosen research technique and its usefulness for this master thesis.

As expressed in the previous paragraph, a generic objective is performed in this study, which is the demonstration of the usability of deep learning methods in industry application. With this goal, the target of the analysis is the development of a new information technology (IT) device through the study of several methods as mathematical proof or proof by demonstration, among others. Once that this is done, the choice of a suitable method for the hand gesture recognition process and the type of connection for the transmission of the information between the different software will be made. An additional and remarkable point is the real time performance of this framework, which should be validated by a build-up of a demonstrator. As a consequence, the research methodology to be used is *vesign and creation strategy*.

This type of research procedures shares its main target with this project, that is the development of new artefacts; hence, to build up IT products. This case of study contains methods and theories that have been used in other studies. In any case, a new domain with the use of ViDi software of Cognex and RobotStudio has been introduced to the application of hand gesture recognition and the corresponding action of the robot. Thus, the application itself is the contribution to knowledge.

Documents, as scientific papers or articles, have been the primary source of information in this project. This data compilation describes information about earlier work in terms of methods or theories that have been used, how the hand gesture image processing should be performed, or the amount of data that is needed for the correct implementation, among others. Moreover, to be supported by former records might increase the reliability of this thesis.

Finally, to approve the proposed solution, the developed demonstrator will be tested and adapted until it is validated.

1.3.2 Process Diagram

In order to achieve the established objectives and aspects, a process diagram (see Figure 1.2) has been designed to display a clear idea of the steps to be followed in the performance of this project.

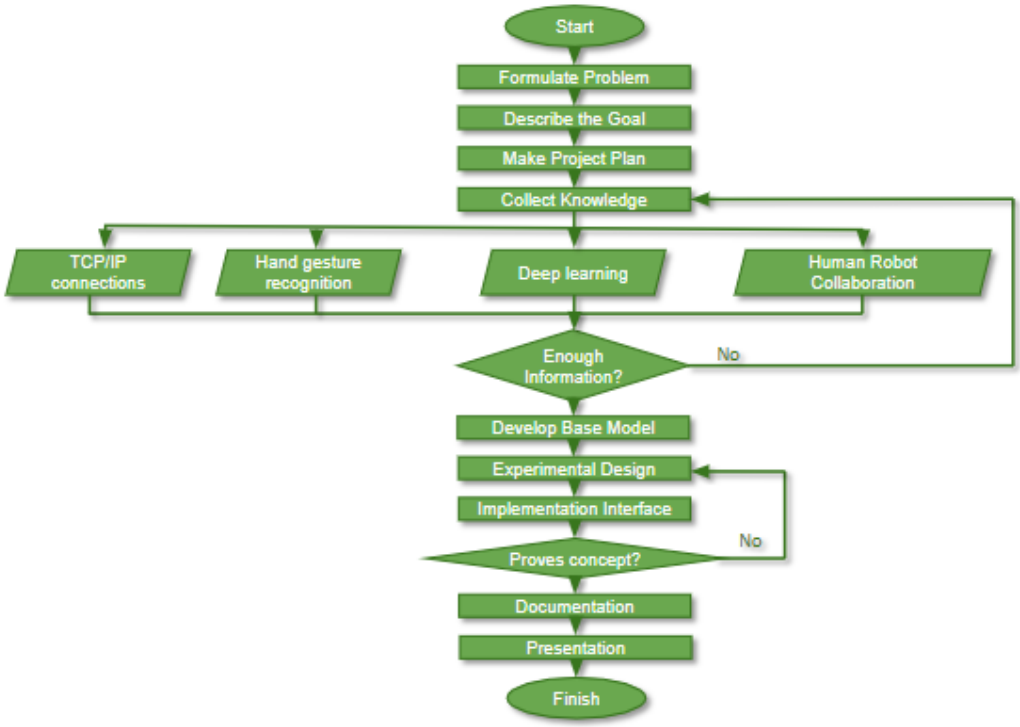


Figure 1.2: Process diagram of the study

Chapter 2

Background

In this chapter the collection of used devices and employed software resources are listed as tools. Furthermore, the state of art associated to the objective of the master thesis is presented.

2.1 Tools

In this section, the set of hardware and software elements used for the development of the project is depicted.

2.1.1 YuMi Robot

For the progress of this project, a robot arm is employed to perform the requested tasks. In this project, the **YuMi** robot by ABB is selected for this aim. ABB is a multinational corporation, whose main business objectives are the electrical energy generation and the industrial automation. In this second sector, their robots are globally well-known.

The YuMi robot (see Figure 2.1), which stands for "you and me", is one of their lastly created collaborative robots in 2015. This dual arm robot, originally has 6 joints in each arm, as most of the robotic arms, but its dual arm disposition gives more reachability and flexibility when it comes to feasible movements and manipulations. This robot is thought to work with humans side-by-side on the same collaborative space. Therefore, safety is built into the functionality of the robot itself. This safety-based architecture allows the robot to perform tasks without the need of a robotic cell surrounding it and still ensures the security of the workers working next to it. The last version of the robot, which is available in single or dual arm, is a 7-axis one, which makes the robot to be more agile than the previous version. In this project, the YuMi robot is in charge of the performance of the tasks and the communication with the user.

Figure 2.1: YuMi robot from ABB

2.1.2 Cognex VisionPro ViDi Software

Cognex corporation is the principal global provider of systems, software, vision sensors and industrial readers used for automation manufacturing processes. The Cogenx's vision software helps companies to improve the quality of their products by eliminating production errors and reducing fabrication costs. The usual applications for this software are defect detection, the supervision of production lines, guideline of assembly robots and the tracking, classification and identification of parts. Because of these applications Cognex software and hardware are commonly used in industrial environments.

Cognex ViDi™ is a deep learning technology which focuses in the analysis of industrial images. This software is able to find specific characteristics, defects and to classify different known types of the same part in a similar way as a supervisor will do in a production line. In case of multiple shape defects, the system is trained in non supervised mode, this way learns the normal shape of the object with its tolerable variations. The software creates a reference model of the object based in this representative images. This is an iterative process which is constantly improving. During this process the parameters can be tuned and the result can be validated until the model works properly. During the execution time, ViDi extracts information from a new set of images and its ANN finds parts, extracts defects and classifies them. This process is defined in Figure 2.2, which deploys two phases, the training phase and the deployment phase.

When it comes to the training phase, the first step is to upload the images to the software. ViDi works with images of high resolution, including colour and thermal images, to recognise virtually anomalies. Then, these images are characterised. To do so, the software is providing four different functions, which are:

- **Locate:** Searches complex characteristics and objects.
- **Analyse:** Detects aspect anomalies and defects.
- **Classify:** Separates and classifies objects or scenes.
- **Read:** Decodes challenging texts and characters.

After the characterisation of the images, these features are used in the training phase. Cognex ViDi works with a small group of images for the training unlike other deep learning based software packages.

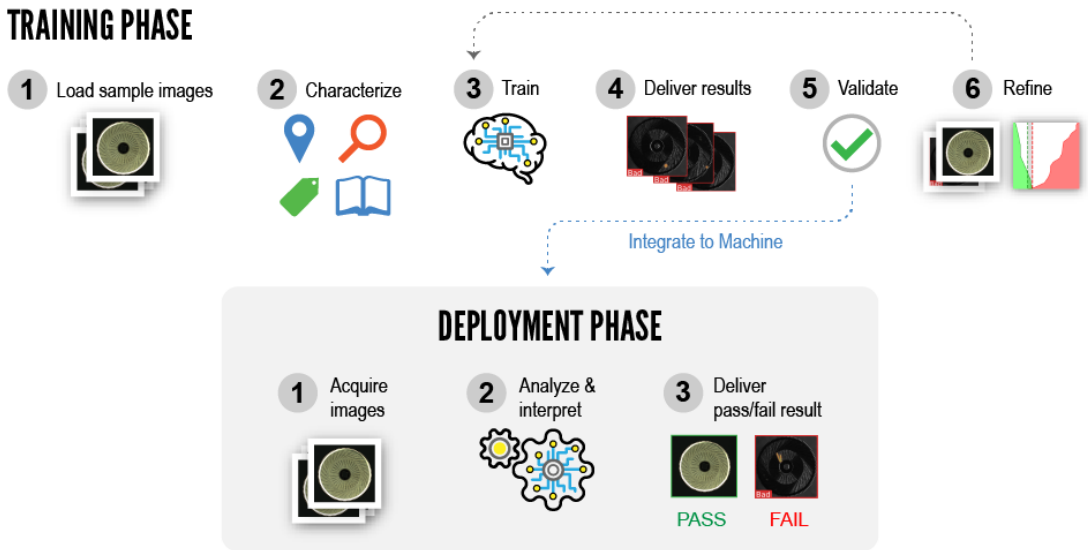


Figure 2.2: VisionPro ViDi software process

Besides, it is able to make complex counting and decodes deformed and difficult reading characters. Once the ANN is trained, the results are delivered in the form of both, a confusion matrix and a ROC curve. In case that results are satisfactory enough this phase would be end. Otherwise, images can be refined to obtain improved results.

2.1.3 RobotStudio Software

RobotStudio is a simulation software for automatic lines of the already mentioned ABB company created specifically for their products. The programming of this software is performed offline to maximise the inversion of robotic systems, this way the production of the line is not interrupted by the programming. This software is constructed in the Virtual controller of ABB, an exact copy of the real software which runs the robot production. This approach allows to perform realistic simulations, with robot configuration folders and programs identical to the ones used in the installations.

To code programs in this software there is a specific language to be used, which is called RAPID. This high level programming language has functions specifically created for ABB robots. A RAPID application is composed by a program and a set of modules of the system. The program is a sequence of instructions which control the robot. This sequence in general is composed of three parts: the main program, the set of sub-routines and the data of the program. This software is going to be used in this project to program and simulate the movements and performance of the tasks of the YuMi robot (refer to Section 2.1.1).

2.1.4 Basler pia2400-17gc Camera

When comes to the cameras there is a wide range of types of cameras designed for specific tasks. As they are 2D and 3D vision cameras, Barcode readers, high resolution cameras, cameras with embedded intelligence and the most common cameras which just take pictures. In this specific case, a Basler pia

2400-17gcv (see Figure 2.3) 2D colour camera of 12 bits of depth with the lens FujiFilm HF9HA-1B is going to be used for the hand gesture recognition.

Figure 2.3: Basler pia2400-17gc camera

2.1.5 Basler acA2500-14gc Camera

When it comes to the camera for the tasks, the Basler acA2500-14gc camera (see Figure 2.4) colour camera will be used with the lens FujiFilm HF9HA-1B.

Figure 2.4: Basler acA2500-14gc camera

2.1.6 TCP/IP Connection

The Transmission Control Protocol (TCP) is one of the essential Internet protocols. It was created between the 1973 and 1974 by Vint Cerf and Robert Kahn. When there are more than one program in the same computer network, TCP can be used to create connections between them so that data can flow. This protocol ensures that data will be delivered without errors and in the proper destiny in the same order that were sent. Moreover, TCP protocol provides the user a mechanism to distinguish different applications of the same machine by the use of "port" concept. TCP is the intermediate layer between the Internet Protocol (IP) and the application (see Figure 2.5).

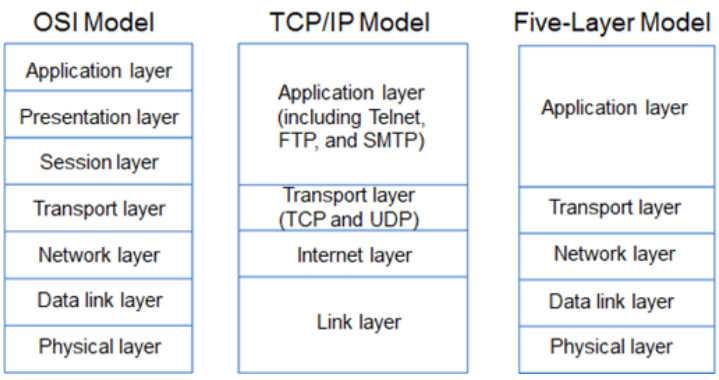


Figure 2.5: TCP/IP model next to OSI

IP is a communication protocol of digital data across network boundaries. Its routing function enables inter-networking, and mainly the establishment of the Internet. Its main function is the bidirectional

use, in origin or destiny of the communication to transmit data by the use of non oriented connection protocols. This communication protocol allows to transfer computed packages across different physical networks which were previously interconnected according to the open system interconnection model, OSI.

2.1.7 Hercules Software

In this project, when it comes to the transference of data from one software to another one will be made by TCP/IP connection (see Section 2.1.6). As far as the different parts of the project, the hand gesture recognition procedure and the robot programming task, are separately developed, a software allowing to test the client/server connection is needed. The Hercules setup utility is endowed with a serial port terminal (RS-485 or RS-232 terminal), UDP/IP terminal and TCP/IP Client Server terminal 2.6. In this case, the TCP/IP client server terminal is used to send messages to RobotStudio 2.1.3 in order to test its socket performance. Eventhough this software was originally made for internal use, nowadays it is a Freeware.

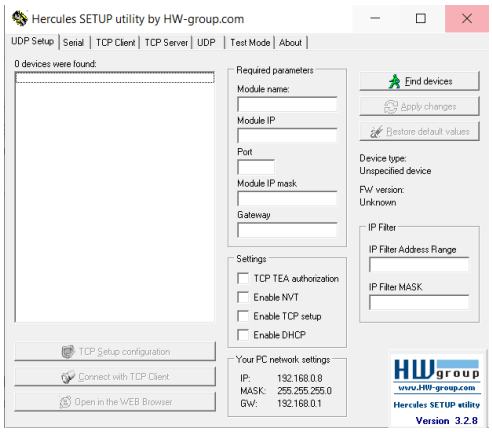


Figure 2.6: Hercules software

2.1.8 EA Vision Studio Software (EAVS)

EA Vision Studio (EAVS) is a Windows based software to develop vision applications for PC. The basic function of this program is to take pictures from cameras, perform image analysis and return measured results (see Figure 2.7). The software enables the configuration of all the input devices to operate the system automatically, which includes cameras, measuring functions, communication, etc. In this project the EAVS will be used first for the capture of hand gesture images in the training phase and after, for the capture of real time hand gesture images and for the application of already trained ANNs.

2.2 State of the Art

In this section, the state of the art of this project is depicted.

Figure 2.7: Examples of EAVS applications

2.2.1 Gesture Recognition

Gesture recognition is a part of the computational science and language technology, its objective being to interpret human gestures by the use of mathematical algorithms. Gestures can be any type of body movement or state, but in general they are performed by the use of hands or the face. Nowadays there are studies about facial emotion recognition and hand gesture recognition [2]. Most of these studies use cameras and vision methods, these are processed by algorithms that allow to interpret gesture language. Moreover, the identification and recognition of the posture, the march, the proxemics and the human behaviour are areas studied by gesture recognition techniques too. Gesture recognition can be taken as a try of computers of understanding the human body language, this way the relation between human and machines will become more strong and easy. This way, old systems as text interfaces or graphical user interfaces (GUI), which still limit the majority of the informative inputs are left behind. Because of all these reasons, the number of gesture recognition applications has had a big increase in the past few years. The different applications of gesture recognition go from human motion recognition [3] to gesture pod for white cane users [4]. Nowadays, to do this type of gesture recognition, there are several methods to obtain the input information as vision based methods [3] which are obtained by the use of cameras, electromyographic [2] and sensors as 3-D accelerometers [5] among others. The most used method for collecting data are vision based methods, which are going to be used in this project. This method requires cameras at specific orientations, this fact makes them impractical in some scenarios as in [4]. Moreover, the input images can be static images (photos) or videos [6], both types of images have been used in several studies related to human body and hand gesture recognition. Furthermore, the images can be taken in different modes as black and white, RGB [6] or RGB-D [7], even though for the recognition of different objects in the image, the RGB or RGB-D images give more information compared with black and white images. Finally, the quality of the image will depend on the number of bits of the image. The needed quality of the image will depend on the purpose of the recognition. For this project, RGB images will be taken with the standard quality given by a mobile phone camera.

Hand Gesture Recognition

Hand gesture recognition refers to the identification of human hand gestures, this is a main part of the previously mentioned gesture recognition area and the focus of this project. In this project, hand gestures will be the input information for the robot, this will allow the user to control the actions of the robot. In general, the hand gesture recognition methods are divided in two groups, methods which use data gloves [8] and the ones that are vision based. On the one hand, the first approach uses sensors that are attached to the gloves which transduces finger movements into electrical pulses to determine hand

gestures. On the other hand, the second approach uses computer vision based techniques, which are not invasive for the user since there is not need of any device attached to the user to transmit the needed information. Moreover this last method works better in controlled environments and not in generic usage processes [9, 10]. In this project, the second method is going to be used, that is to say vision based techniques.

Once the recognition method is selected, the hand gesture taxonomies and representations should be taken into account. Therefore, it is important to select the type of image, static [11] or dynamic [7], which depends in the type of information that is wanted to be processed. The static hand gestures are described as the position and orientation of a hand image while a specific amount of time in which there is not any movement. On the other hand, the dynamic hand gestures includes movements associated to body parts as it is waving a hand [10]. Another used method is the Motion Fused Frame application, which allows to recognise movements from a group of static images which are correlated between them, because each image is part of the same hand movement [12]. This method is used to enable movement recognition with static images, this way dynamic images are not needed. For the implementation of this project, static images are enough besides the available camera does not allow to take video recordings. Due to these facts, static pictures are going to be taken and used for the hand gesture recognition.

When it comes to the representation of the gestures, there are two main categories; 3D model based and appearance based representations [10, 13]. Each of them have different recognition techniques for gesture representation as 3D textured volumetric [5], which contains highly detailed human body skeleton and skin surface, 3D geometric model that has less information about the skin surface but still contains skeleton information and 3D skeleton model, that is a really simple skeleton representation [14]. For these techniques one or more cameras are focused on the real target. On the other hand, the appearance based repressions have different techniques too, which are colour based models that uses colour based markers, silhouette geometry model which contains several geometric properties (perimeter, convexity, surface, etc.), deformable gabarit model and motion based model which uses the motion of an object for the recognition [12].

For the recognition of the hand gestures there are different methods or algorithms as K-means, whose main objective is to decide k points that are centres to minimise the clustering error. Another used algorithm for the vision recognition is the k-nearest neighbour which classifies objects referenced on closest training examples in the feature space. Mean shift clustering is a non-parametric technique, this method does not need prior knowledge of the number of clusters and the shape of the cluster is not constraint. Another used method is Support Vector Machine (SVM) which is a non-linear classifier, this technique in general obtains better results than previously mentioned methods [10]. The Hidden Markov model (HMM) [15] can be considered as a generalisation of the Markov chain method but without its chain limitations, this method that was introduced in the mid 1990s became quickly recognised [16]. The Dynamic Time Warping [17] algorithm takes two possible points out of two signals depending in their associated feature values and calculates the distance between them. In addition to the previously mentioned methods, the Time delay neural networks is a special Artificial Neural Network (ANN) whose main focus is to work with continuous data making the architecture of the Neural Network (NN) adaptable to each case, this is highly advantageous for real time applications. This method is commonly considered as an extension of multi-layer perceptron. To finish, the Finite State Machine [18] method has a limited number of possible case scenarios, an infinite machine could be created but would not be practical.

The hand gesture recognition in this project will be in real time and with several different users who have not been tracked before. For this aim, the best method to be used is the deep learning method as mentioned before, specifically Convolutional Neuronal Network (CNN). There are several projects that used CNNs for the hand gesture recognition in the past years [12, 14, 19]. These articles show the good performance of CNN for hand gesture recognition with high accuracy percentages, which validates the use of this method for this specific procedure.

2.2.2 Artificial Neuronal Network

Besides the previously mentioned hand gesture recognition, in this project another two tasks should be performed by ANN. For both tasks, images are taken to lately use as image input of the ANN and get the results of them. Due to this, ANN for computer vision purposes will be used to classify and process the taken images. In [20], different networks for vision purposes are explained, as Amari and Hopfield Networks. On the one hand, Amari network has been studied since 1960 and in 1970 proposed two self-organizing random networks. The first network is a non-recurrent network for association and the second one is a recurrent network used for concept formation. On the other hand, the Hopfield network is a discrete network which uses two state threshold neurons. In [20], next to [21] show the applicability of ANN in vision problems.

The purpose of each task is different. In task 1, scratches or errors of a industrial piece are wanted to be found and classify the industrial piece as good or bad. To do so, small differences in the surface of the industrial piece should be found out by the ANN. In [22] the same principle is used, to find for anomalies in the image. In this case, ANN searches for diverse medical anomalies in different images as cervicovaginal smears, breast cancer histopathology and in ultrasound imaging of the carotid artery. For the classification task of these images an ANN composed by Self-Organizing Map (SOM) is employed. In [23], errors are searched for errors in the eggshell by colour based error detection by the use of ANN. Specifically, the trained ANN founds out blood spots in the eggshells. If the eggshell does not have any blood spot, is classified as A type egg whereas if the eggshell has blood spots the egg is classified as defective. For this application, authors used a commercial NN named NeuroShell^{TM} with an accuracy of 92%. For the task 1 recognition methods could be used too as in [24, 25]. In task 2, dices which have images of muffins and chihuahua dogs are thrown inside a box, the objective of the ANN is to classify correctly the images of the dices and to count them. There are many articles that talk about classification of images by the use of ANN [26, 27, 28]. In [27], the objective was the quality inspection of beans based in their size and colour. For that, authors created a hardware, a software based in Matlab and the images were processed as RGB images. By the created software a mean accuracy of 90.6% was obtained and the used method to obtain this result was ANN. Another example of the use of an ANN for classification is [28], where weeds and crops are classified. To do so, a back propagation ANN is created which can distinguish young corn plants from weeds. For that 40 images of each type were trained in the ANN. The obtained results were 100% of success in the classification of corn plants and a 80% of success in the classification of weeds. Which are good results taking into account the amount of used image database. These articles show that ANN are a good option for the two tasks performance, where anomalies search and classification methods could be applied.

2.2.3 HRC – Human Robot Collaboration

Mechanical devices have been used since ancient times. There are several concepts similar to robots found approximately 400 years BC. For example, the famous inventor Leonardo da Vinci from 1495 had drawings of a mechanical knight. Even though, the word *robot* was first used by the Czech writer Karel Čapek in a 1921 play. The word comes from a Czech word, *robota* that means work or compulsory work. The first collaborative robot or *Cobot* was invented in 1996 by J. Edward Colgate and Michael Peshkin. They call Cobot to a device and a methodology that enables physical direct interaction between a person and a manipulator controlled by a computer. With the years many different cobots have been commercialised by different companies as Kuka or ABB [29].

The robots, which are mainly known as robotic arms that work in structured factories are moving to human populated environments, this makes a need of more complex cognitive abilities in robots. The efficiency and safety characteristics of robots are not enough for human populated environments, they need to be more cooperative and communicative with humans. The HRC field has a wide range of applications, coming future and potential economical impact [30]. To add these characteristics to the robots, the ability of performing usual tasks in human environment, one of the most used methods is the deep learning techniques. This way, the robot is able to learn from its environment and adjust depending the specific requirements [31]. When it comes to the introduction of Cobots in industrial environment, where they have to coexist with other robots and human being the security systems and design considerations change. What are known as industrial robots need security barriers to ensure the safety of the workers in the facilities, whereas cobots do not need any barriers because of their internal security mechanisms. This internal mechanisms are force detectors which in case the robotic arm feels a force against itself, the motors of the joints of the arm are stopped. Even though, the supply of the robotic arm is not switched off, so the robotic arm still has available all the connections with other devices and the electricity source. Many articles talk about these specifications and purpose safe human-robot collaborative workplaces design considerations [32, 33].

Furthermore, even though still cobots are more used in laboratories than industrial environments, this number is increasing and there are several studies that support the idea of using cobots in industrial environments. One of the main worries in industrial environment to use this type of robots is the security concerning as no physical barriers are used to protect the workers from the movements of the robot. Nevertheless, this insecurity regarding to the security issues is decreasing and companies are feeling more comfortable when it comes to the use of cobots in industries [34]. This is due to the different applications and studies done with cobots around this topic. For example, in [35] the usability of cobots is enhanced for middle sized volume tasks and the possibility of investing in small dynamic and flexible work stations that can easily be changed of work stations. This way, in case a specific part of the line needs an extra help, the cobot station can be moved an reprogrammed easily saving time and money to the company. Another main worry around cobots or robots in general is the fear of unemployment in between the workers. For this reason it is important to define and assign correctly the tasks of the production line. Robots are great for repetitive, uncomfortable or even dangerous situations for humans, because robots unlike humans they do not feel pain or fatigue. So, if the cobot is correctly placed in the production line and helps in the performance of the human worker, the human being will not feel threaten by the robot [36]. There are other many studies about the application of cobots in industrial environments [37, 38].

In this project, a collaborative robot is used which performs tasks in collaboration with human beings.

To do so, the previously mentioned security specifications and characteristics of the robot will be taken into account.

2.2.4 Dual Arm Robot Programming

In this project, the YuMi robot of ABB is going to be used (see Section 2.1.1) which as it can be observed in Figure 2.1 is equipped with two robotic arms that are attached one to the other one by a common base. The robots with this composition are called dual arm robots. This configuration allows the programmer to design synchronised motions or master-slave configuration in each arm, where one arm movements are defined by the movements of the other arm [39]. These kind of motions often are quite complex to program since each arm is controlled independently. This happens in the YuMi robot too, even the exiting physical connection of both arm, each robot arm program is independent. So, to synchronise both arms, there are specific functions created in RAPID code for RobotStudio ABB software. There are two ways to synchronise the arms; by synchronisation point where both arms work independently but one arm waits the other one until this reaches the corresponding point and by synchronisation motion where both arms start and end the motions in the same time. In case of synchronised motions, master-slave configuration can be settled which allows to configure the position of one arm relative to the position of the other arm [40].

The YuMi robot has been used in several human-robot collaborative projects as making a puzzle [41], gift wrapping [42] or inserting a lid on a toy box [40]. These articles show the usability of this robot in synchronised tasks which have human interaction. This project is performed in Sweden during the Spring period, which means the weather conditions which affect the robot conditions have to be taken into account as mentioned in [42].

Chapter 3

Demonstrator Setup

In this first phase, all the different setup and layout selections are explained. Moreover, the selection of the hand gestures and the robots actions, the reasoning of the layout of the work-space and the selection of the components needed for the correct operation of this project are performed.

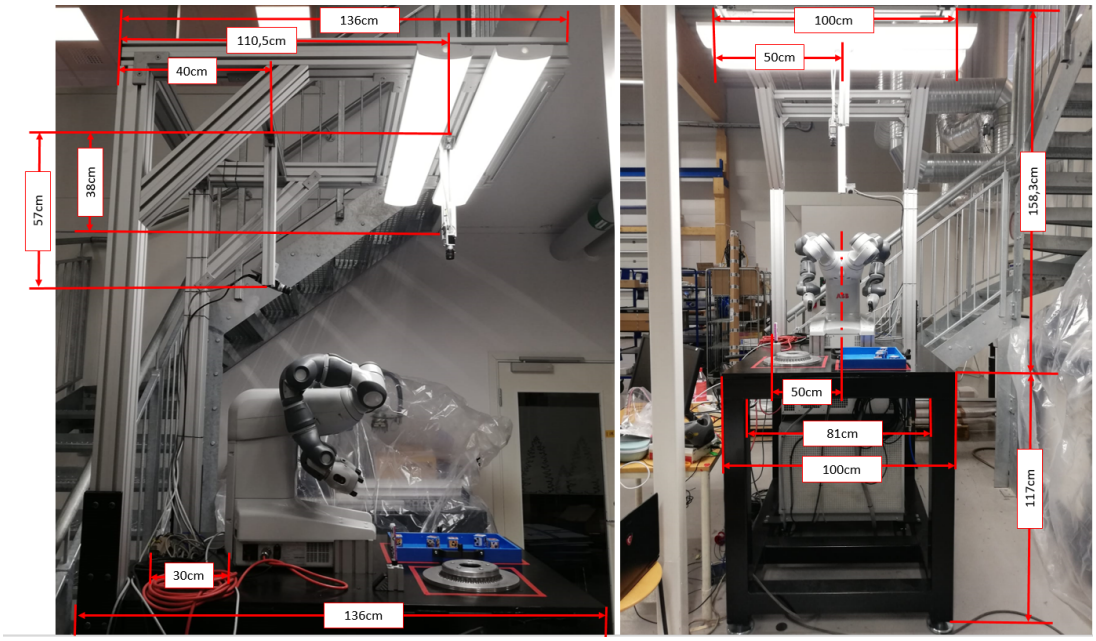


Figure 3.1: Structure with measurements

3.1 Layout Settings

For the correct performance of this project, a structure has been mounted. In this section, the different taken decisions and details of the setup of the project are reported. To place the different parts that correspond to the two tasks of the project and the YuMi robot itself a metallic table, which in the company is used for training purposes, has been chosen. To this metallic table a structure has been added, which is used to place the two needed cameras for the extraction of the images and the lights for the control of

the ambient light. The position and size of these lights has been crucial for the correct illumination of the parts of the tasks. The positions of all the mentioned objects can be observed in the Figure 3.1.

Moreover, the size of the lights positioned in the two sides of the task camera is of 1200 × 200mm. Smaller lights were tried out but the reflection of the industrial part was unavoidable. Therefore, the ANN trained to search faults in the industrial part, incorrectly detected reflections as errors.

When it comes to the cameras, the camera used for the hand gesture detection is the Basler pia2400-17gc camera (see Figure 2.1.4) and the camera to take the images of both tasks is the Basler acA2500-14gc camera (see Figure 2.1.5).

Moreover, for a good performance of the YuMi robot when it comes to holding the pen in Task 1 or the dices box in Task 2, grippers and handles specifically designed for this project have been used. The grippers shape has been designed for an easy and accurate grip of the pen (see Figure 3.2). Furthermore, the shape of the handles of the box have been designed according to the specified grippers adding them a slot for a better fastening of the box (see Figure 3.3). Finally, to hold the pen correctly a pen holder has been manufactured, as it can be observed in Figure 3.4.

Figure 3.2: Designed grippers

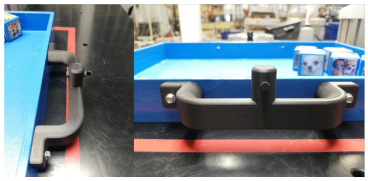


Figure 3.3: Designed handlers

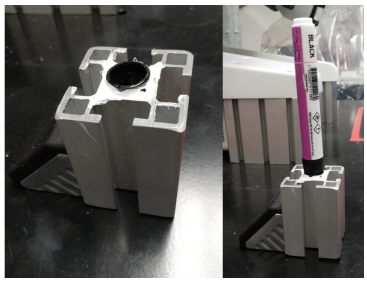


Figure 3.4: Designed pen holder

3.1.1 Task 1: Industrial Part

The main objective in Task 1 is to detect failures on an industrial part. A breaker of a car with a highly reflective protective layer has been chosen (see Figure 3.5). Besides, a permanent black pen is selected to write markings in the previously mentioned industrial part.

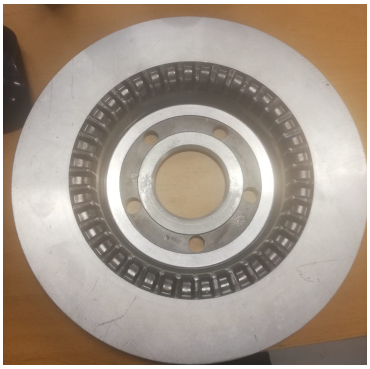


Figure 3.5: Selected industrial part

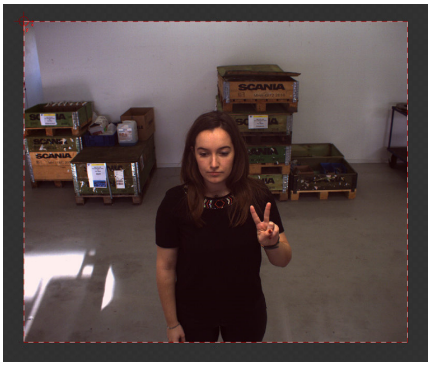


Figure 3.6: Selected ROI for ANN classification

3.1.2 Task 2: Chihuahua Muffin Dices

To setup the dices for the Task 2, the images printed in Figure 3.7 were placed in wooden made dices as it can be observed in Figure 3.8. For that, 15 different images of muffins and 15 different chihuahua images have been used. These images have been extracted from two different free online databases[1].

Figure 3.7: Chihuahua vs muffin photos database

Figure 3.8: Chihuahua vs muffin dice setup

3.2 Hand Gesture Selection

In this section, the research, reasoning and selection of the used hand gestures is going to be explained. For the selection of the hand gestures, elements like cultural aspects, ergonomics, simplicity and common knowledge have been taken into account. Each hand gesture should be easy recognisable and related to a common global meaning. The user, without a reference manual, should be able to know the meaning associated to the hand gesture. To get reliable information about different hand gestures around the world some articles and web pages have been checked [43, 44, 45].

In the implementation of this project, two stages have been defined. In each stage, a different task will be performed. To tell the robot which task to perform, a hand gesture will be linked with each task. Hence as there are 2 tasks the easiest way to communicate with the robot which of them is the selected is by the use of number hand gestures (NHG). According to [45] the western Europe hand gestures combined with the hand gestures used in The United States (US), United Kingdom (UK), China and Sweden have been selected, as far as the Japanese hand gestures have been discarded (see Figure 3.9). This selection

[1] `http://www.image-net.org/synset?wnid=n02085620` and `https://github.com/bfolkens/animals_or_food`

is due to the fact that this project was thought to be exposed in a technological fair in Sweden where the majority of the users are expected to be European, English speakers and specially Swedish. Besides, the Japanese NHG use the opposite logic to all the other cultural NHG's. That is to say, in European countries the number of shown fingers is the number which is wanted to be expressed, whereas in Japan the number of hidden fingers is the number that is wanted to be transmitted.

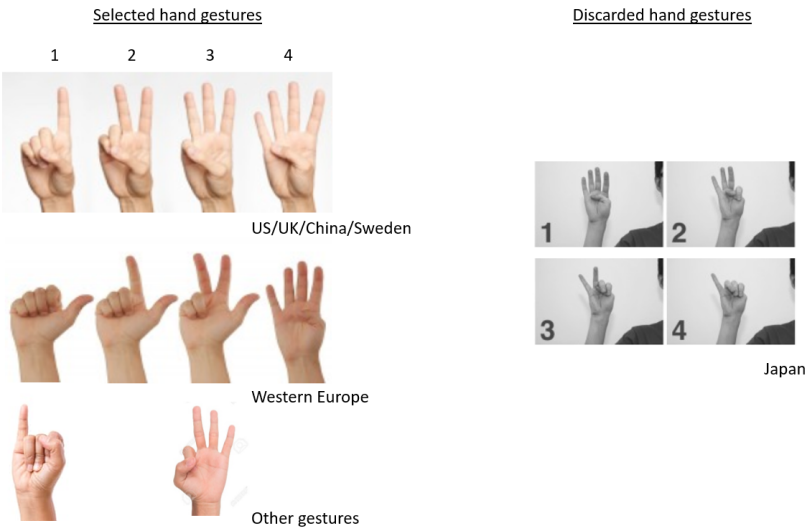


Figure 3.9: Selected and discarded number hand gestures

Beyond the NHG, other gestures would be added to get a more complete control of the robot, which are *stop*, *ready*, *give* and *cancel* gestures (see Figure 3.10). The *stop* gesture allows to stop/freeze the robot when it is moving according to a previously selected movement. While the *stop* gesture is recognised, the robot will remain stopped; when the hand is taken out from the camera, that is to say the *stop* gesture is not being recognised anymore, the robot will resume the previously calculated movement. In case that the robot is moving to a selected task (*1*, *2*, *good*, *bad*, *middle*, *give* or *rock*) and the task is wanted to be changed, then *cancel* gesture would have to be performed. This way, the selected task will be erased and another task can be selected. When it comes to the *ready* gesture, this will be used when the user wants to communicate to the robot that users interruption in the task has finished and the robot can continue with its work. Last but not least, the *give* gesture informs the robot to give an advertisement present to the user.

Figure 3.10: Other selected hand gestures:*stop*, *rock*, *cancel*, *middle*, *bad*, *good*, *give* and *ready*.

Because the final purpose of this project is to be placed in a technological fair workshop, its main

objective is to be attractive for the visitors in the fair. With this aim, five additional gestures have been added which are *good*, *bad*, *middle* finger and *rock* gestures (see Figure 3.10). These gestures do not activate a task or process, they just make a movement in the dual robot arm which express happiness, sadness, madness or the movements of playing a guitar (see Figure 3.11 and Figure 3.12). This way, the workshop will be more interactive and entertaining for the visitors of the fair.

In total, there would be 16 different gestures, some of them with the same meaning, this makes the user to have more options when it comes to the communication with the robot. To summarise, the selected hand gestures would be the following ones:

- **One:** UK, US, China and Sweden style (Figure 3.9).
- **Two:** UK, US, China and Sweden style (Figure 3.9).
- **One:** Western Europe style (Figure 3.9).
- **Two:** Western Europe style (Figure 3.9).
- **One:** Other case scenarios (Figure 3.9).
- **Stop** (Figure 3.10)
- **Bad** (Figure 3.10)
- **Ready:** Making a circle with the index finger and thumb (Figure 3.10).
- **Good:** Closing the hand and opening the thumb up (Figure 3.10).
- **Cancel:**With the hands in fist mode (Figure 3.10).
- **Cancel:**With the hands opened (Figure 3.10).
- **Give:** Opening the palm up (Figure 3.10).
- **Rock:** Showing the small and index fingers (Figure 3.10).

3.3 Selection of Final Actions

In this section, the actions to be performed by the robot are going to be analysed and selected depending on the input hand gestures explained in Section 3.2.

As explained in Section 3.1, the project will have two different tasks, in each task a vision recognition process will be performed. To inform the robot which of these tasks should be executed, the number hand gestures will be used (see Figure 3.9). Hence, when the made hand gesture is a number, the robot will move to the corresponding task and will perform the expected task.

3.3.1 Task 1: Find Errors on the Metallic Part

If the user makes the number *one* gesture (see Figure 3.9) is selecting to do the first task which is to find errors, for example scratches, in a metallic piece. In this task, first the robot will pick the marker and will move it in the air so that the user proceeds to make markings in the metallic part. Once the gesture is done, the robot will give the marker to the user. The user will have to take the marker and make some paintings in the metallic part. Once the marks are done, the user will have to leave the marker in the support of the marker and make the *ready* gesture. This way, the robot understands that the drawing of the piece has been finished and can take a picture of the metallic part to process the image and find the painted marks. Once the markings are found, the industrial piece with the detected markings will be shown in a screen. For the search of the marks, an ANN will be created.

3.3.2 Task 2: Play Yatzy with Chihuahua and Muffin Photos dices

If the user makes the number *two* gesture (see Figure 3.9), the second task is selected, which is to play Yatzy with chihuahua and muffin dices. Yatzy is a really popular game in the Scandinavian countries and it is played with 5 dices. The main objective is to obtain 5 dices with the same number, this achievement is called "yatzy". To obtain 5 dices with the same number, the player has three throws. After each roll, the player chooses which dice to keep, and which to re-roll. In this case, spite of numbered dices, the faces of the dices will be covered with photos of chihuahuas and muffins. The objective of the player is to get 5 chihuahuas. To do so, the player has three throws. Each time that the player throws the dices, the robot will take a picture of the dices and count the amount of chihuahuas and muffins. In case there are less than 5 chihuahuas and the player has thrown less than 3 times, the robot will allow the player to throw again. Otherwise, if the user has rolled the dices three times and has not obtained 5 equal dices the robot will make the sad gesture. Conversely, if the player has obtained 5 equal dices, within three throws, the robot will make the happy gesture.

3.3.3 Emotion Robot Gestures

Whereas, if the introduced hand gesture is the *good* gesture (Figure 3.10) the robot will do the happiness movement represented in Figure 3.11. The same happens if the *bad* gesture (Figure 3.10) is made. In this case, the sadness movement will be performed by the robot (see Figure 3.11). If the user shows the middle finger to the robot, this will get mad and will answer the user with a boxing movement (see Figure 3.12). To finish, if the user makes *rock* gesture, the robot will get excited and make the movement of playing a guitar (see Figure 3.12). These movements are predefined.

3.3.4 Robots Reactions to Stop, Cancel, and Give Gestures

In case the *stop* gesture (Figure 3.10) is performed in front of the camera, the robot will freeze the movement that is currently been performed. When the hand is taken out from in front of the camera, the stop gesture is not recognised anymore, the robot will continue with the movement that was being

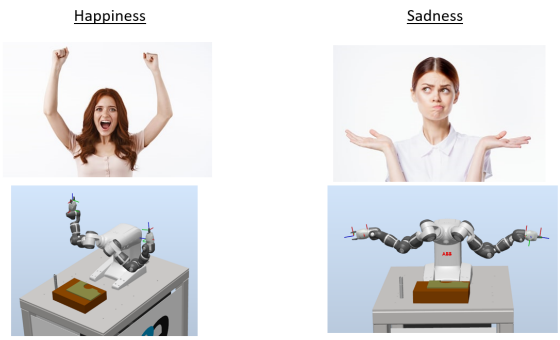


Figure 3.11: Happiness and Sadness robots reactions

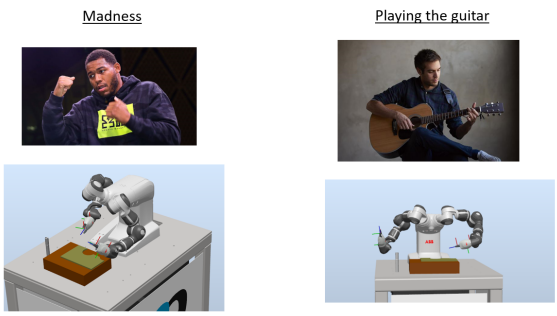


Figure 3.12: Madness and Rock robots reactions

performed before the stop of the movement. Hence, the stop gesture allows to block the movement or trajectory of the robot while the stop gesture is being performed.

If the user makes the *cancel* gesture (Figure 3.10), the outcome of this gesture is to cancel the previously selected task. Furthermore, first a task has to be selected to be able to use this gesture. For example, the user selects the first stage task by making the gesture of *one* (Figure 3.10) and then decides to perform another task, so the user cancels the selected task by performing the *cancel* gesture (Figure 3.10). This gesture cleans the previously selected task path and moves the robot to the home position waiting for the selection of another task. Moreover, *stop* and *cancel* gestures should be introduced after the selection of another task. The separate selection of these tasks does not make sense since they allow to modify or control the movement of the robot.

The last gesture is *give* gesture (Figure 3.10), which asks the robot to give something to the user. As this project is thought to be for a fair, the robot will give an advertisement of the company EA as present to the user, when this gesture is performed by the user.

3.4 User Process Explanation

To understand the available tasks for the user and the correspondent actions performed by the robot the flow chart in the Figure 3.13 has been generated.

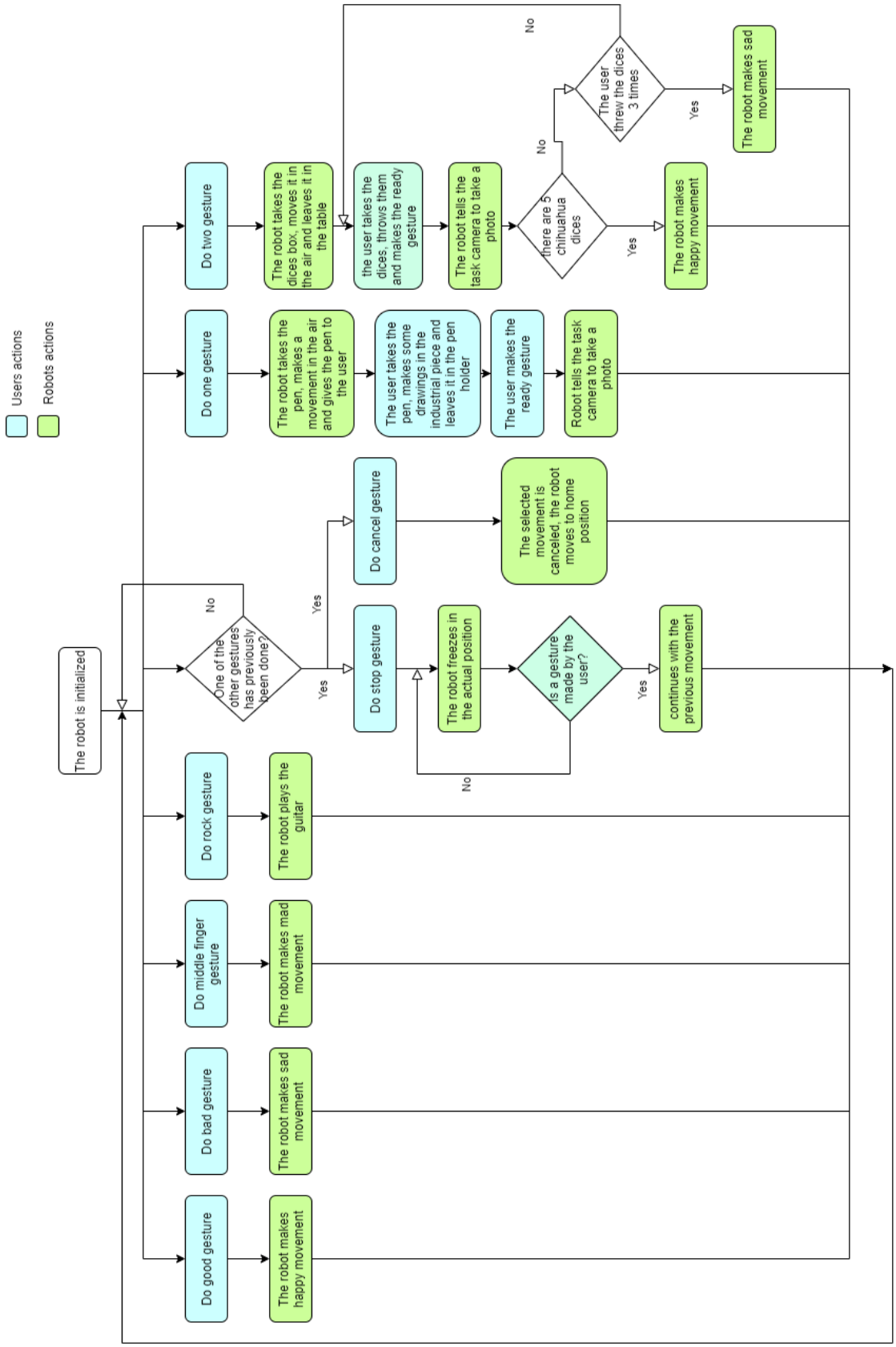


Figure 3.13: The flow chart of the process to be followed by the user

3.5 Photo Database

In this section, the followed process to obtain needed photos for the training of the ANN is explained. To take the images needed to train the three ANN's, two cameras, Basler pia2400-17gc (Section 2.1.4) and Basler acA2500-14gc camera (Section 2.1.5) have been used. The Basler pia2400-17gc camera has been used for the hand gesture task and the second camera, the Basler acA2500-14gc camera, for the other two tasks. The two cameras are placed in the middle front of the table as described in Section 3.1. For the control of the two cameras in the different situations EAVS program has been used (see Section 2.1.8) and three different recipes have been created. One recipe was made for the hand gesture camera with a exposure time of 15.000μ s, the other two recipes were created with the Basler acA2500-14gc camera. One of the recipes in this camera is for Task 1 with a exposure time of 7.000μs and the other recipe is for the Task 2 with a exposure time of 35.000μs. The difference in the exposure time of the recipes for the different tasks is depending the amount of light that is needed for each task. In case of Task 1, the industrial piece has a tendency to reflect the light really easily, because of this the exposure time of task one is much smaller than the exposure time of task two. A lower exposure time means that the picture taken by the camera is darker, this way an image with less reflections in the industrial piece is obtained, due to this is easier to process the photo.

3.5.1 Hand Gesture Images Database

To get a well trained ANN, photos with different hand types, sizes and skin colours are needed with both left and right hands, making the different gestures defined in Section 3.2.

To obtain this data base of images, the camera has been set in the same position as in the final setup (see Section 3.1). The used camera is Basler pia2400-17gc with the lens FujiFilm HF9HA-1B which allows to take colour images and focus the image to a certain distance. A colour camera was selected spite of white and black camera to have more information to train the ANN. Since the hand is more easily differentiated from the face of the person by the use of colour cameras. To obtain the needed photos for the development of this project, the EA Vision Studio (EAVS) software (Section 2.1.8) has been used, which is a vision software created by the company EA. This software allows to obtain, trait and process the images.

The needed images have been taken during two days to different people in the company. Each person has performed all the gestures introduced in Section 3.2 with both hands and in both sides of each hand. Fourteen people have participated in the image obtaining phase and 588 photos have been taken. The amount of photos which correspond to each hand gesture can be observed in Table 3.1.

images for each gesture	one	two	stop	bad	ready	good	cancel	middle	rock	give	person stands still	no one in photo
session 1	186	123	31	30	34	30	45	31	48	30	0	0
session 2	338	274	109	125	170	114	45	109	152	122	41	7

Table 3.1: Number of images taken in each photo session

After the tests of the ANN with the 1st photo session data base, the need of more photos was realized.

To prove the capability of the location tool, some photos with people standing still and without people were taken too. In the end, the number of photos listed in the 2nd photo session were obtained (see Table 3.1).

3.5.2 Task 1 Images Obtaining

In this subsection the created image database for the Task 1, industrial part task, is reported. As the objective of this task is to find errors or scratches in an industrial part, two types of photos have been taken. On the one hand, pictures of the industrial piece without any type of marks have been taken (see Figure 3.14). On the other hand, photos of the industrial part with different types of markings, which have been done with a black pen, have been taken (see Figure 3.15). In the created database for the Task 1, there are 21 images of the part without marks and 59 images of the piece with marks. The reason to take more images with marks than without, is that the markings made by the user can vary a lot.

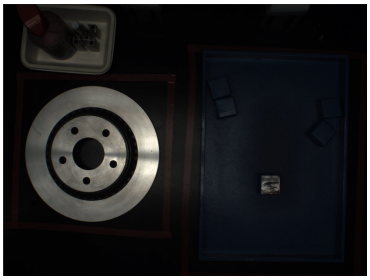


Figure 3.14: Task 1 piece without markings

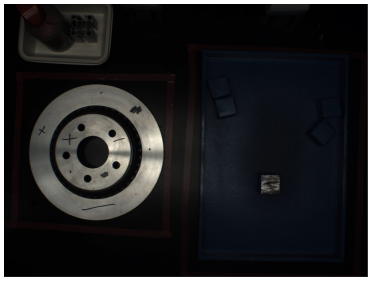


Figure 3.15: Task 1 piece with markings

3.5.3 Task 2 Images Obtaining

For Task 2, images of the dices placed in different positions and with different rotations withing the box have been taken. For the training of the ANN, 59 images of the dices have been captured, an example of it can be observed in Figure 3.16.

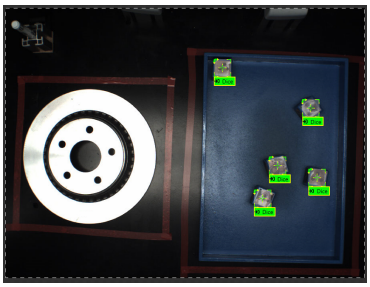


Figure 3.16: Task 2 image with location tool applied

If the Figure 3.15 is compared with Figure 3.16 the difference in the lighting between the two images is observable. As expressed before in Section 3.5 the brightness of the images is related to the settled exposure time of each recipe.

Chapter 4

The Training phase

In this second phase, all the needed steps to get a trained ANN for the recognition of hand gestures, the search of markings in the industrial piece and the classification and counting of muffin and chihuahua dices will be listed. These steps can be observed in Figure 4.1.

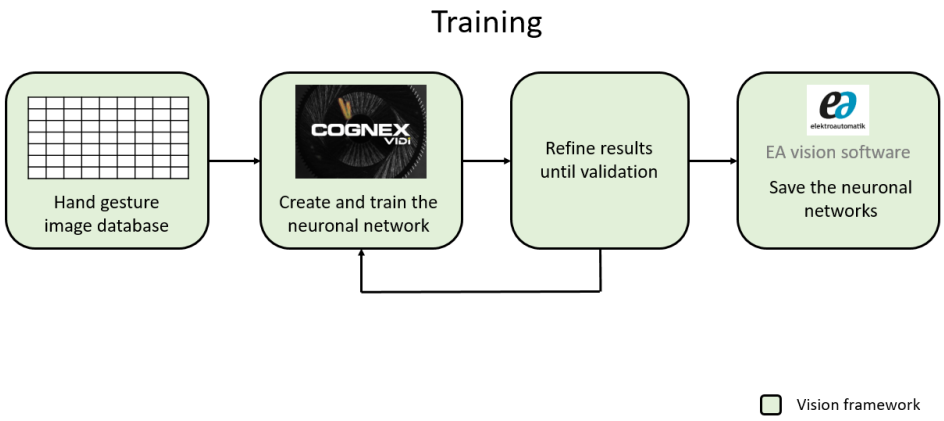


Figure 4.1: Workflow of the training phase

To do so, first the hand gestures that are going to control the robot have to be decided. Next, the ANN is designed and trained. Once the ANN gives the first results, these are analysed and, if needed, refine them until acceptable results are obtained.

4.1 ANN ViDi Results Interpretation

The ViDi software, when the ANN is designed and run, gives the obtained results in form of a table in case there is one class, a ROC-AUC curve in the case there are two classes and a confusion matrix in case there are more than two classes (see Table 4.1).

In this project, when the Location tool is used there will be only one class, the hand. Hence, the

	Table	ROC curve	Confusion matrix
number of classes in Classify	1	2	>2
number of classes in Locate	always	–	–

Table 4.1: ViDi software methodology to display the results

obtained results from the program will be a table which represents the number of found hands, trained photos, labelled photos, recall, precision and F-score.

The recall or sensitivity is the ratio of the total number of correctly classified positive examples divided by the total number of positive samples. If the recall is higher means that the class is correctly recognised. Its equation can be observed in Equation 4.1, where the True Positive (TP) is an outcome where the model correctly predicts the positive class and False Negative (FN) is an outcome where the model incorrectly predicts the negative class,

$$Recall = \frac{TP}{TP + FN} \tag{4.1}$$

The precision or positive predicted value refers to the data classified correctly over all the data that has been classified as positive in each type of gesture, in this case. Its equation is similar to the recall equation, but spite of taking all the photos of the current class, it takes all the photos that have been predicted as the current class. That is to say, the TP is divided by the sum of TP and False negative (FP) (see Equation 4.2), that refers to the images that have been incorrectly classified as the current class.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

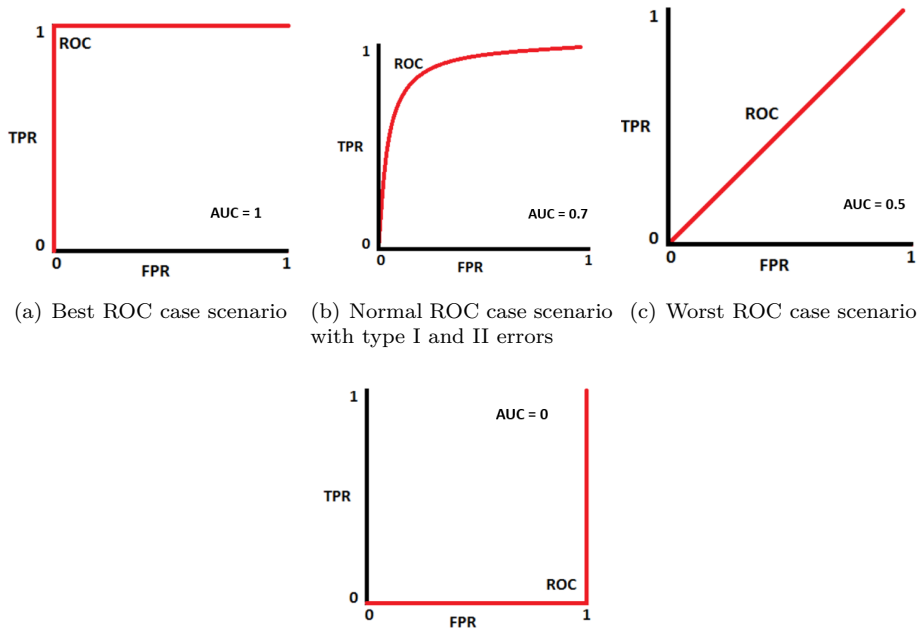
To finish, the F-score is the combination of the recall and the precision and it is calculated as expressed in the Equation 4.3,

$$Fscore = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.3}$$

In the classification tool, the number of classes will depend in the number of gestures that have been uploaded to the work-space. In this project, the number of gestures used in the work-space will change while the ViDi software is being tested. Once conclusions are obtained, the last ANN will be created with all the gestures commented in Section 3.2. So, in general, results will be obtained by the information given in the confusion matrix.

4.1.1 ROC Curve

When there are two classes defined in the ANN, the results are given by the ROC curve (see Figure 4.11) and the AUC (area under the curve) value, apart from the precision, recall and the F-score values. The ROC curve is a probability curve that displays the values of the recall (Equation 4.1), otherwise called true positive rate (TPR), against the false positive rate (FPR). An ROC space is defined by FPR and TPR as x and y axes, respectively. The FPR is calculated by the Equation 4.4.



(a) Best ROC case scenario (b) Normal ROC case scenario with type I and II errors (c) Worst ROC case scenario

(d) classes confused with each-other

Figure 4.2: ROC curves

$$FPR = 1 - specifity = 1 - TNR = \frac{FP}{TN + FP} \tag{4.4}$$

Furthermore, the AUC is the rate that expresses the ability of the model to distinguish between positive and negative classes, hence the ability to classify different classes. In general, the AUC rate is shown next to the ROC curve (see Figure 4.2).

4.1.2 Confusion Matrix

When there are more than two defined classes to train the ANN, the results are given by the confusion matrix (see Figure 4.3). Apart from that, the precision, recall and the F-score values are expressed. If the Figure 4.3 is observed, the diagonal that goes from the left up corner to the right down corner represents the classes that are correctly classified. All the circles that are out from this diagonal are miss-classified pictures. As it can be observed in the previous image, the main diagonal remarked in green has the biggest circles which means most of the photos are correctly classified. The circles that appear out from the confusion matrix, the ones remarked with a red square, are the ones that have not been classified at all. This means that there was not a class outstanding from the others. This is calculated by the percentage of similarity of the current image compared with other learnt gestures. Moreover, the confusion matrix provides the needed information about which gestures are miss-classified with which ones and the level of miss-classification in percentages. For example, in Figure 4.3 is remarkable that the *two* gesture is highly confused or miss-classified with *one* gesture and vice versa.

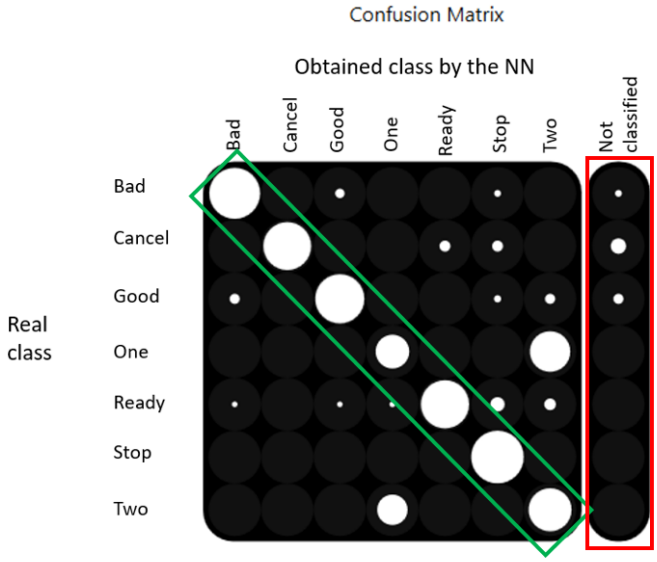


Figure 4.3: Confusion matrix explanation

4.2 Hand Gesture ANN

4.2.1 Structure Creation of the ANN

For the hand gesture recognition, ViDi software has been used, which has 4 different tools for the creation of the ANN that can be combined to get different results (see Section 2.1.2). After reading the capabilities of the different tools, the Reading and Analyzing tools were discarded. On the one hand, the Reading tool's main objective is to read codes, words, or serial numbers which for hand gesture recognition does not have any usability. On the other hand, the Analyzing tool finds error in the images of the work-space, this can not be applied for the hand gesture recognition neither. So, the remaining tools, Location and Classification tools, will be tested to see which one performs better to reach the goal of this project. In this section, the different tested ANN structures are going to be stated.

Testing of Different Structures

As explained before, two different tools, the Location and Classification tools, will be tested to see which one is the best option to get best results for the hand gesture recognition. To have equal conditions in the test of these two different tools, both will be tested with the same pictures, same amount of photos, same Region Of Interest (ROI) and the same percentage of trained images. The photos will contain two hand gestures, *one* number hand gesture and *two* number hand gesture (see Section 3.2).

When it comes to the training of the images, the 70% of the images will be randomly selected. This selection will be done with three of the structures and the results will be compared between them to see which one obtains better performance. For this testing, 309 photos are added to the work-space. From that 309 photos, 186 are *one* number hand gestures and the remaining 123 are *two* number hand gestures.

Classification Tool Structure

As it was stated in the beginning, the performance of the different tools of the ViDi software to recognize hand gestures was completely unknown, hence different structures have been tried. In this section, the use of only the Classification tool will be reported (see Figure 4.4).

Figure 4.4: Setup of the classification ANN

Following all the assumptions explained in Section 4.2.1, 70% of the uploaded photos have been randomly trained in the classification ANN. To run the Classification tool, each photo is labeled with a class, in this case *one* or *two*. The obtained result from this training can be observed in Figure 4.5. As it can be seen in the results, the ANN was not able to learn due to the amount of information in each picture. This is caused by the selected ROI, as the photos are taken with the person in the middle of the image surrounded by the actual environment, the tool is not capable to understand what is important in the photo. Hence, the tool takes all the information of the image as samples and gets lost (see Figure 4.6).

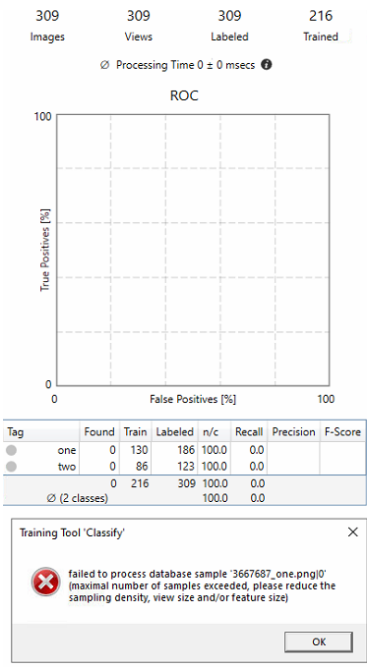


Figure 4.5: Results of the tested classification ANN **Figure 4.6:** Selected ROI for classification ANN

In conclusion, the Classification tool itself can not be used for the hand gesture recognition since the photos of the user will be taken as expressed in Figure 3.6.

Location Tool Structure

In this second test, the Location tool itself is used as the structure of the ANN. As in the previous case, specifications explained in Section 4.2.1 are followed. Hence, 309 photos were added to the work-space and the same ROI as in Figure 4.8 was taken for this test. Even though, as in the Location tool an

identification window (see Figure 4.8) is used to locate the part that is wanted to be analyzed, some images were erased as some of them were blurry or confusing even for a human. Because of this reason, a total of 297 photos were in the work-space, where 178 were labeled as *one* and 119 were labeled as *two*. The labelling of the photos is allowed by the identification window, which gave the possibility of naming each window in each photo as a new class. In this case, the identification windows will be named with two different classes, *one* and *two*. The obtained results from this experiment, after training the 70% of each class, can be observed in Figure 4.7.

297 Images | 297 Views | 297 Labeled | 207 Trained

∅ Processing Time 18.4 ± 7.5 msecs

Feature	Found	Train	Labeled	Recall	Precision	F-Score
one	173	124	178	66.7	73.5	69.9
two	117	83	119	58.3	61.8	60.0
	290	207	297	63.3	68.7	65.9
∅ (2 classes)				62.5	67.6	65.0

Figure 4.7: Results of the tested Location ANN with 70% training

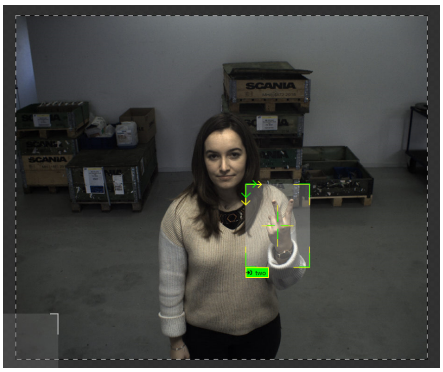


Figure 4.8: Selected ROI and identification window of Location ANN

As it can be observed in Figure 4.7, 173 images of *one* label were found over the existing 178 and in case of the *two* label, 117 were found over 119. But this does not mean the found images are correctly classified. Since the precision and recall of the images is so low, the miss-classification of most of the images can be concluded. Hence, the precision is 73.5% in the case of class *one* and 61.8% in case of the class *two*, meanwhile recall is of 66.7% and 58.3%, respectively. The Location tool is not useful to classify the different labels. Moreover, most of the hands are found, 290 over 297 photos, so the Location tool works properly when it comes to the search and localization of hands in the images.

Location plus Classification Tools Structure

After observing the results and obtained conclusions of the Classification and Location tool, the best way of approaching hand gesture recognition is throughout a combination of both tools (see Figure 4.13). The same work-space used in the Location tool structure has been used in this test. That is to say, a total of 297 photos were in the work-space, where 178 were labeled as *one* and 119 were labeled as *two*. In this case, the Location tool will only be used to locate the hand in the photo. Due to this, all the identification windows will be named as hand (see Figure 4.10) so that any classification is given at this stage of the structure of the ANN. Hence, in the Location ANN there will be only one class named *hand*.

As it can be observed in Figure 4.10, most of the hands were found in the 297 labeled photos, 293 to be exact, with a precision value of 100% and a recall value of 95.6%. All this result with a randomly selected 70% of the total photos (207) for the training of the ANN. Taking these figures into account, a high efficiency when it comes to the localization of the hand can be concluded. These results compared with the ones explained in Location tool structure are much more encouraging.

Once the hand is found within the image by the use of the Location tool, the Classification tool

297 Images | 297 Views | 297 Labeled | 207 Trained

∅ Processing Time 17.6 ± 3.3 msecs

Feature	Found	Train	Labeled	Recall	Precision	F-Score
hand	293	207	297	95.6	100.0	97.7

Figure 4.9: Results of the tested Location ANN with 70% training of the Location + Classification structure

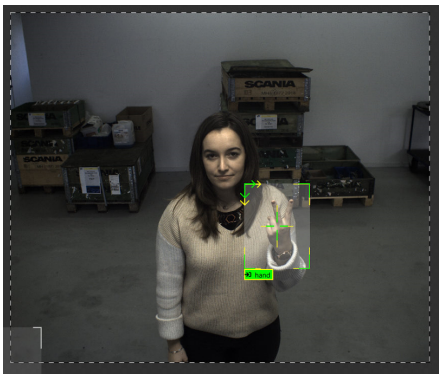
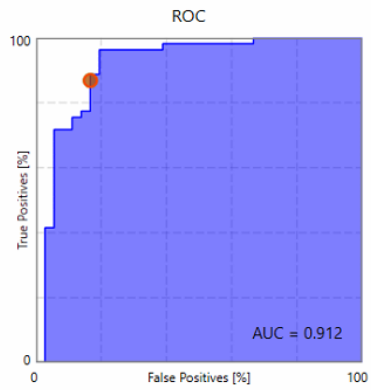


Figure 4.10: Selected ROI and identification window of Location ANN in the Location + Classification structure

will identify the type of gesture that is being performed by the user. As the Location tool created the identification window observable in Figure 4.10, the ROI of the Classification tool will be limited to the identification window as can be seen in Figure 4.12. This means that the Classification tool will have a smaller image, less information, to be processed. This way, the problem that appeared in the Classification tool structure in Section 4.2.1 is avoided. So, once the ROI is applied to the images, these are labeled as *one* or *two* depending in the gesture observable in the image. As in all the previous cases, this ANN will be trained randomly with the 70% of each class photos getting the results in Figure 4.11.

297 Images | 293 Views | 293 Labeled | 204 Trained

∅ Processing Time 16.1 ± 0.9 msecs



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
one	174	122	175	0.0	86.8	88.5	87.6
two	119	82	118	0.0	83.3	81.1	82.2
	293	204	293	0.0	85.4	85.4	85.4
∅ (2 classes)				0.0	85.1	84.8	84.9

Figure 4.11: Results of the tested Classification ANN with 70% training of the Location + Classification structure

Figure 4.12: Selected ROI and identification window of Classification ANN in the Location + Classification structure

In this case, the Classification tool does not give any error of the number of data sample, instead gives the ROC curve and the AUC value next to the precision, recall and F-score values. As it is observable in Figure 4.11, the ROC curve is near of getting a perfect result, a complete fulfilled square with an AUC of 1. This is appreciable in the values of the recall, precision and the F-score too, which are over 80% of performance. If these values are compared to the ones obtained with the Location tool structure, it is remarkable the achieved improvement. For example, when the *one* label has to be classified the

precision in this case is of 88.5%, whereas in case of the Location structure this value decreases in a 15% obtaining a precision of 73.5%. The value with the biggest difference between this test and the Location tool structure test is the recall of the label *two*, in this test takes a value of 83.3% whereas in the Location structure takes a value of 58.3%, this means a decrease of 25%.

In conclusion, the best structure for the hand gesture recognition is the combination of the Location and Classification tools. First, applying the location tool for the search of the hand in the picture and then the classification tool to classify the type of gesture in the processed image.

Final ANN Structure

To perform the main objective of the ANN, that is the recognition of hand gestures, two steps have been completed. First the recognition of the hand over the photo has been performed and then, once the location of the hand is established, the type of gesture performed by the user has been classified. To do so, locate and classify tools of ViDi software have been used in that order (see Figure 4.13). First in the input box, the images are added to the work-space. This box is just for the uploading of the needed images.

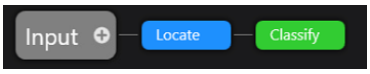


Figure 4.13: Setup of the final ANN

Once the images are available, the Location tool that itself is a ANN will locate the hand in the image as it can be observed in Figure 4.14.

Figure 4.14: Locate hand gestures

As it can be observed, some of the added images (see Figure 4.14) have people standing still without doing any hand gestures or non people at all. In these cases, the ANN does not recognise any hand in the picture, what means that the pictures that are not labelled, do not have any hand, are not going to

be processed by the next tool, the Classification tool. Furthermore, in the first two photos, where the *one* and *stop* gestures can be observed, the ANN finds the hands and remarks them.

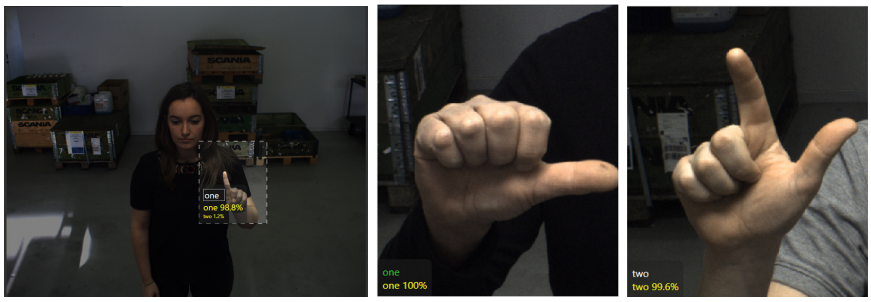


Figure 4.15: Classification of hand gestures

In the previous images(see Figure 4.15), the behaviour of the classify tool can be observed. First, the region specified by the location process is taken over all the image, only the information that is inside this region is processed by the Classification ANN. If the middle and right images in Figure 4.15 are observed, the different colours of the words one and two can be seen. These colours represent whether the image is in the training set or not, the name in green means that the image is being trained and in white that the image is taken in the test set.

Furthermore, the fact that these ANN tools are black boxes must be also remarked. This means that the user is not able to tune any parameter of the ANN as weights or structure. Hence, to understand the behaviour of these ANN it is important to test different case scenarios and see how the black box system reacts.

4.2.2 Training of the ANN

In this section, the training of the ANN's for the Location and Classification tools is explained as well as the analysis of the obtained results with different data sets. As explained before, the Location and Classification tools are black boxes, hence there is not any available information about the structure or the weights in them. Due to this fact, different proves with different data sets have been performed. Starting from simple ones, with few gestures, and finishing with the complete set of photos with all the gestures expressed in Section 3.2. For all these tests, the final ANN structure will be used, that is the combination of Location tool with Classification tool.

First Approach: Workers Photos with Basic Gestures

In this first approach, the photos of 14 workers of the company have been used, with the 6 basic gestures that are *bad*, *good*, *stop*, *one*, *two* and *ready*. With this input data set, first the Location ANN has been trained to locate the hand in each photo. Once the location of the hands was performed, the Classification of the obtained photos has been performed.

Location Tool As explained before, the location tool is the responsible to find the hand in the image. To do so, in this first approach 356 images have been used as data set. From that 356 images the 80% has been used to train the ANN, that is to say 284 photos. With this data set the obtained results are expressed in Table 4.2.

Feature	Found	Train	Labelled	Recall	Precision	F-score
hand	355	284	356	98.6	100.0	99.3

Table 4.2: 1st approach: Results of the location tool

As in this case there is only one feature to be found in the images, *hand*, a confusion matrix is not plot in the program. In its place, the Table 4.2 is obtained. In this table, the number of trained photos, labelled photos and found photos is observable. If the number of figures in the table are observed, the labelled photos are 356 and the found hands are 355 what means that an image has not been found. Because of this failed photo, the recall is of 98.6%, the percentage in which the class is correctly recognised, while the precision is 100.0% and the F-score 99.3%. These values show a really good performance of the ANN. Moreover, all the photos used in this first approach have people making gestures. This means that there is not any photo without people or people without doing anything, so for the ANN is easier to find the hands in the photos.

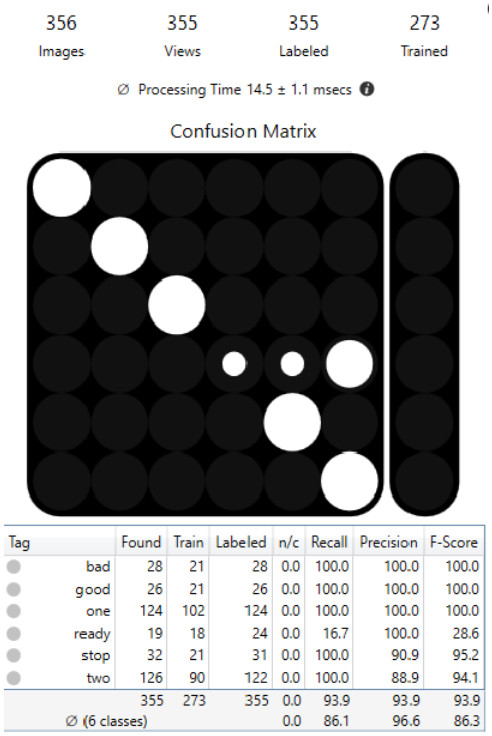
Classification Tool In this part of the first approach, the classification of the previously cropped images will be performed with a data set of 356 photos and 6 different gestures. In the *one* gesture, the little finger gesture has been deleted to avoid mismatches in the training and to have a more balanced data set. Apart from that, when the photos were taken to the volunteers, some of the volunteers expressed the difficulty and feeling uncomfortably when they were performing this gesture. Due to all these facts, this gesture has been deleted.

To train this ANN, first 50% of the total photos has been trained randomly. Once the results of the 50% trained photos was obtained, the miss-classified photos where added to the training set until a 70%-80% of each gesture photos was trained. This way, the results are presented in Figure 4.16

The successful training percentages of this approach are the following ones:

- **One:** 102/124 →82% trained.
- **Two:** 90/122 →73.77% trained.
- **Stop:** 21/31 →67,74% trained.
- **Bad:** 21/28 →75% trained.
- **Ready:** 18/24 →75% trained.
- **Good:** 21/26 →75% trained.

As it can be observed in the results, the obtained precision and recall are really good in the *bad*, *good* and *one* gestures, *stop* and *two* gestures have an acceptable performance but the *ready* gesture recall is



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	28	21	28	0.0	100.0	100.0	100.0
good	26	21	26	0.0	100.0	100.0	100.0
one	124	102	124	0.0	100.0	100.0	100.0
ready	19	18	24	0.0	16.7	100.0	28.6
stop	32	21	31	0.0	100.0	90.9	95.2
two	126	90	122	0.0	100.0	88.9	94.1
	355	273	355	0.0	93.9	93.9	93.9
∅ (6 classes)				0.0	86.1	96.6	86.3

Figure 4.16: 1st approach: Results of the Classification ANN with 70% training

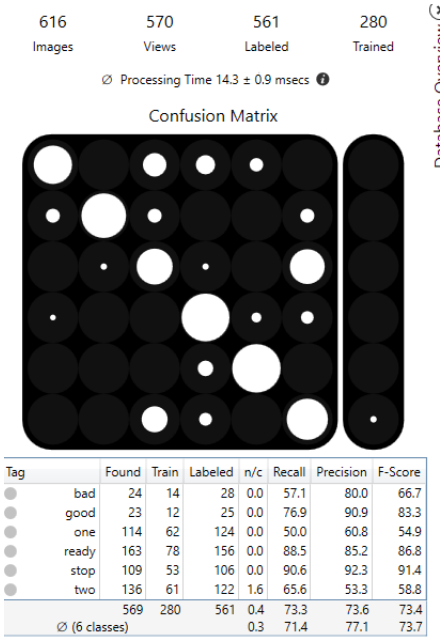
really bad. In the case of the *ready* gesture, there are more miss classified photos than correctly classified, this being unacceptable. This problem can be caused by the unbalanced number of photos between the different pictures since *one* and *two* gestures have more than double photos to be trained and tested. Even though, a clear reason of why the *ready* gesture is miss-classified in such percentage can be obtained just looking these results. This is due to the ignorance of the structure and the way the weights are settled in the ANN of the classification tool. This uncertainty makes the writer think about different possibilities to get these results. One possibility could be that the ViDi software gives the weights to the classes depending in the amount of photos that has that class, in that case the *one* and *two* gestures would have higher priority than the others. Another possibility is that the ANN needs more information to be able to identify the *ready* gesture without any confusion. On the one hand, the solution to the first option is to add a balanced number of photos to every class/gesture. Hence, to have a similar amount of photos in all the gestures for the training and testing of the ANN. On the other hand, the solution for the second option is to add more photos to the gestures that are being confused, *ready* and *stop*, and have less images than *one* and *two* gestures.

Second Approach: Adding Images of *ready* and *stop* Gestures

In this second approach, more photos for *ready* and *stop* gestures have been added to the original data set, so that the confusion regarding to these two gestures is minimized, as explained in the previous test. As the training of the Location tool ANN does not really change, the only difference is the number of the data set which will increase, the analysis of the Location tool is not going to be performed in this section. On its place, the results of the classification tool will be analysed to see whether the increase in the amount of available images of the *ready* and *stop* gestures makes a difference in the classification of the different classes. Apart from that, the indirect effect in the results of the other gestures will be

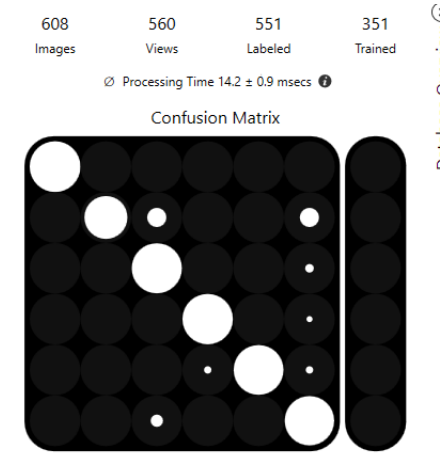
evaluated.

Classification Tool As explained previously, more photos of *ready* and *stop* gestures have been added to the original data-set, this way a work-space with 561 photos has been obtained. First, 50% of the photos was randomly trained and the results of it can be observed in Figure 4.17. As the main objective of this test was to see the effect of adding more photos to the most miss-classified gestures, the obtained results will be compared to the ones achieved in the previous test. A remarkable fact is the acquired improvement in the recall of the *ready* gesture, which increased in a 71.8% (see Figure 4.17). Even the results of the *ready* gestures are improved, the other gestures have worse F-score than in the previous case. Moreover, these results are obtained with the 50% of photos used for training of the ANN. So, the miss-classified photos will be added to the training set until a 70-80% of the photos are used for training. This way, a similar situation to the previous one with better results will be obtained (see Figure 4.18).



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	24	14	28	0.0	57.1	80.0	66.7
good	23	12	25	0.0	76.9	90.9	83.3
one	114	62	124	0.0	50.0	60.8	54.9
ready	163	78	156	0.0	88.5	85.2	86.8
stop	109	53	106	0.0	90.6	92.3	91.4
two	136	61	122	1.6	65.6	53.3	58.8
	569	280	561	0.4	73.3	73.6	73.4
∅ (6 classes)				0.3	71.4	77.1	73.7

Figure 4.17: 2nd approach: Results of the Classification ANN with 50% training



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	28	20	28	0.0	100.0	100.0	100.0
good	21	16	23	0.0	71.4	100.0	83.3
one	127	90	124	0.0	97.1	91.7	94.3
ready	160	86	156	0.0	98.6	98.6	98.6
stop	102	53	100	0.0	95.7	100.0	97.8
two	122	86	120	0.0	94.1	88.9	91.4
	560	351	551	0.0	96.0	96.0	96.0
∅ (6 classes)				0.0	92.8	96.5	94.2

Figure 4.18: 2nd approach: Results of the Classification ANN with 70% training

Results are listed as,

- **One:** 90/124 →72.58% trained.
- **Two:** 86/120 →71.67% trained.
- **Stop:** 53/100 →53% trained.
- **Bad:** 20/28 →71.43% trained.
- **Ready:** 86/156 →55.13% trained.
- **Good:** 16/23 →69.56% trained.

With the listed values of training percentage, the obtained results have improved significantly (see Figure 4.18) compared with the previous case scenario. That is to say, the performance of *ready* and *stop* gestures have increased unquestionably. But, even the *bad* gesture still has a really good recall and precision, the *good* gesture has low recall. Furthermore, the precision and recall of the different gestures is not really affected because of the addition of the *ready* and *stop* gesture images. Otherwise, all the precision and recall values of all the classes will change and that did not happen with the *bad* gesture. So, if the behaviour of the ANN is taken into account can be said that the weights are not related to the amount of images of each label. Moreover, as the *good* gesture recall has decreased, more images will have to be added to get better results.

Third Approach: Balanced Number of Photos with Basic Gestures

In this third approach, more photos of *good* and *bad* gestures have been added to the previous data-set so that the number of photos of each gesture is similar. Furthermore, the possibility of weight assignment depending in the number of photos of the class is further tested. Apart from the added hand gesture photos, some other photos with anyone on them or people without doing any hand gesture were added (see Figure 4.14). This way, the capability of the Location tool to locate hands in different case scenarios is tested.

Location Tool In this third approach, 785 images have been used as data-set. From that 785 images, among the photos that are labelled (737 photos), 80% have been used to train the Location ANN, that is to say 590 photos. With this data-set the obtained results are expressed in Table 4.3.

Feature	Found	Train	Labelled	Recall	Precision	F-score
hand	736	590	737	98.6	98.6	98.6

Table 4.3: 3rd approach: Results of the Location tool

Even photos without people and photos with people without doing any hand gestures (see Figure 4.14) were added, the performance of the Location tool ANN continues to be really good with a recall and a precision of 98.6%. This can be observed in the number of found photos over the number of labeled photos because the number of found photos is one less than the number of labelled ones. This means, only one photo of the hand gestures was not found. Apart from that, it should be noted that photos that are not labelled (the ones that they do not have any hand gesture) cannot be trained. This means that the added photos of either, a person standing still without doing any hand gesture or the photos where there is not any person, are not trained. Hence, the ANN was able to identify correctly the photos where a hand was visible.

Classification Tool In this part of the third approach, the classification of the previously cropped images will be performed with a data-set of 737 photos and 6 different gestures. In this case, all classes have similar amount of photos in the data-set, what makes a more balanced training of the ANN.

To train this ANN, first 50% of the total photos have been trained randomly (see Figure 4.19). Even

the results obtained with this percentage of the training data-set were pretty good, more images were added to the training set to get more precise results with higher recall values. Once the results on 50% trained photos were obtained, the miss-classified photos where added to the training set until a 70%-80% of each hand gesture photos were trained. When the ANN was trained with approximately 70% of the data-set, the number of total labeled images was smaller, this can be observed in Figure 4.19 and Figure 4.20. The number of labelled photos is reduced to 731, 6 photos less than in the previous case. This is due to the difficulty to understand the hand gesture in some photos, these images were not significant for the ANN, so they were erased. This way, the results presented in the Figure 4.20 were obtained.

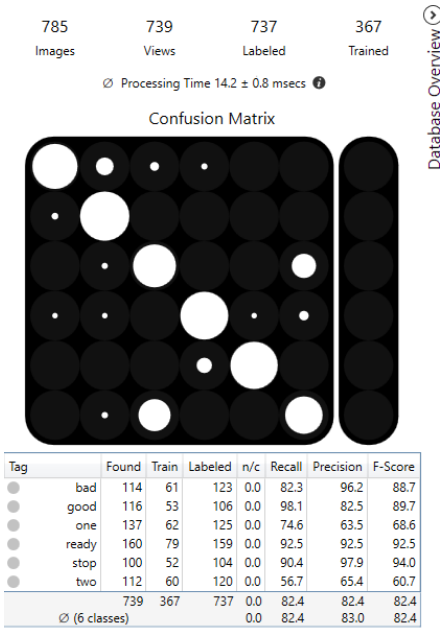


Figure 4.19: 3rd approach: Results of the Classification ANN with 50% training

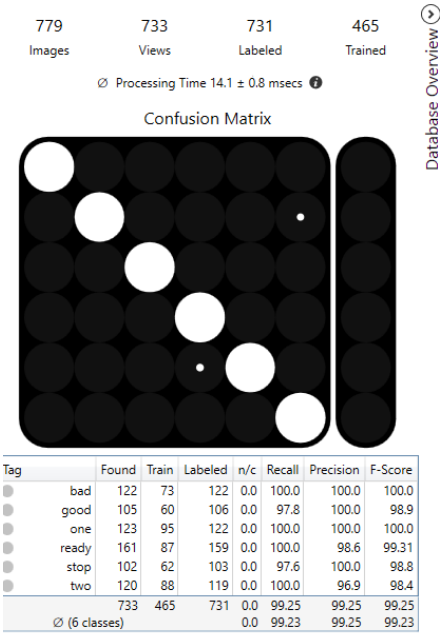


Figure 4.20: 3rd approach: Results of the Classification ANN with 70% training

The trained percentages of the 70% approach (Figure 4.20) are the following ones.

- **One:** 95/122 →77.87% trained.
- **Two:** 88/119 →73.95% trained.
- **Stop:** 62/103 →60.19% trained.
- **Bad:** 73/122 →59.84% trained.
- **Ready:** 87/159 →54.72% trained.
- **Good:** 60/106 →56.6% trained.

If the results obtained in Figure 4.19 and Figure 4.20 are compared, the improvement of the precision and the recall can be observed. In the confusion matrix, the miss-classified images are clearly represented, which are the circles that are located out from the diagonal that goes from the left up corner to the right bottom corner. The confusion matrix in Figure 4.20 has less circles out from this diagonal than the one

located on Figure 4.19. Hence, the confusion matrix located to the right has classified the different classes better than the one located to the left. This is due to the amount of images that have been used for the training set in each case.

The results obtained after the 70% of the training of the photos are really good since all F-score values are over 98%. The lowest value is the precision for the *two* hand gesture which takes a value of 96.9%, that is a very good precision value. Hence, the results obtained after this experimentation are applicable for the hand gesture recognition with good predicted results. Even though, more gestures (*cancel*, *rock*, *give*, *middle*) are missing in the training of the ANN. When this missing gestures are added to the work-space a change in the performance of the ANN and the results is expected.

Fourth Approach: Adding *cancel* Gesture to the work-space

Once the behaviour of the ANN when it comes to the weights is concluded, the effect of the class size or amount of information in the image is wanted to be tested. This is due to the selection of one of the gestures, the *cancel* gesture, which needs a bigger identification window than the other gestures.

Location Tool As said before, the *cancel* gesture takes a bigger area in the image, hence a bigger identification window is needed (see Figure 4.21). This makes the Location tool to train with a bigger amount of information in the window.

Figure 4.21: Difference in the size of *cancel* and other gestures

The size given to this identification window is the same for all the photos that are being used in the work-space. Hence, the identification window is fixed for all the gestures, once the size of the window is selected cannot be modified depending on the type of gesture. This means that for *cancel* gesture, a bigger window will be optimal but this will affect the other gestures since their window will grow too, getting not significant information for the ANN.

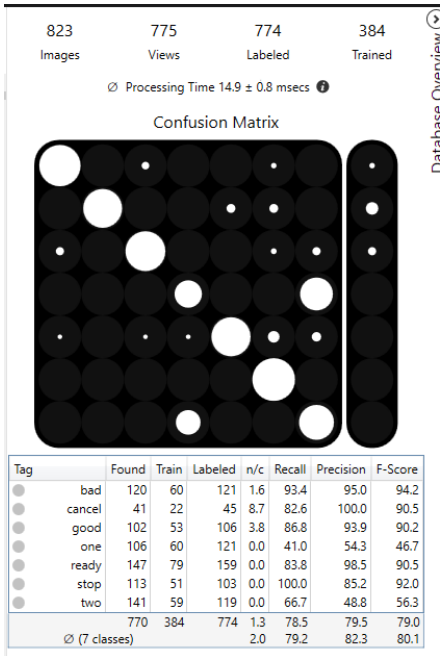
Feature	Found	Train	Labelled	Recall	Precision	F-score
hand	775	620	775	97.4	98.1	97.7

Table 4.4: 4th approach: Results of the Location tool

Even though, looking to the results obtained in the first running of the location tool (Figure 4.4), the good performance of the hand location is not affected. From 823 pictures, 775 are labelled, which

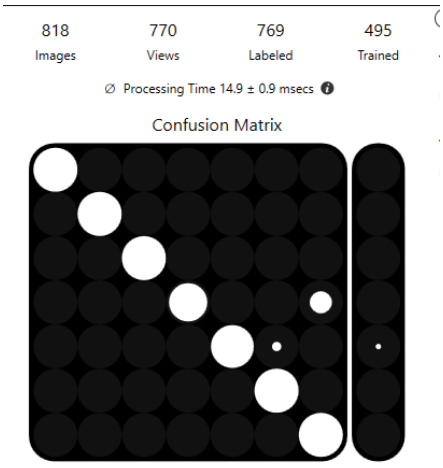
means the rest are people standing still or pictures without any person in front of the camera (48 photos). Among the 775 photos that are labelled, 620 are trained in the ANN, that is 80% of the photos. With this amount of photos a recall of 97.4% and a precision of 98.1% is obtained. The fact that all the labelled photos were correctly found is remarkable as well as the good performance of the Location ANN. So, in conclusion, the change in the size of the selection window of the Location tool does not affect the performance of this tool.

Classification Tool As the identification window has been changed for the incorporation of the *cancel* gesture to the gestures group, the amount of information that is available for the Classification tool will be affected. As the ROI of the classification tool becomes bigger, this tool will have more information to learn for each gesture. To see how this change affects to the results of the ANN, the same steps followed in the two previous cases have been accomplished. Firstly, 50% of each gesture data-set has been trained obtaining the results displayed in Figure 4.22.



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	120	60	121	1.6	93.4	95.0	94.2
cancel	41	22	45	8.7	82.6	100.0	90.5
good	102	53	106	3.8	86.8	93.9	90.2
one	106	60	121	0.0	41.0	54.3	46.7
ready	147	79	159	0.0	83.8	98.5	90.5
stop	113	51	103	0.0	100.0	85.2	92.0
two	141	59	119	0.0	66.7	48.8	56.3
	770	384	774	1.3	78.5	79.5	79.0
∅ (7 classes)				2.0	79.2	82.3	80.1

Figure 4.22: 4th approach: Results of the Classification ANN with 50% training



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	121	67	121	0.0	100.0	100.0	100.0
cancel	43	27	45	0.0	100.0	100.0	100.0
good	105	69	105	0.0	100.0	100.0	100.0
one	112	94	118	0.0	75.0	100.0	85.7
ready	157	92	159	1.5	94.0	100.0	96.9
stop	106	59	102	0.0	100.0	93.5	96.6
two	125	87	119	0.0	100.0	84.2	91.4
	769	495	769	0.4	96.4	96.7	96.5
∅ (7 classes)				0.2	95.6	96.8	95.8

Figure 4.23: 4th approach: Results of the Classification ANN with 70% training

If the results of the previous tests are observed, a tendency of miss-classification between the *one* and *two* gestures can be seen, which with the adding of the *cancel* gesture becomes more obvious. As still a 20% of each class can be added to the training data-set, the mismatched photos of each gesture will be added to this set until it contains between 70-80% of the photos of each hand gesture. This way, the ANN has more information since the miss-classified photos are used to train the ANN. Moreover, this makes an improvement of the precision and recall of the confusion matrix as it can be observed in Figure 4.23.

Results are listed in the following points.

- **One:** 94/118 →79.66% trained.
- **Two:** 87/119 →73.11% trained.

- **Stop:** 59/102 →57.84% trained.
- **Bad:** 67/121 →55.37% trained.
- **Ready:** 92/159 →57.86% trained.
- **Good:** 69/105 →65.71% trained.

If precision and recall of the different classes in Figure 4.23 are observed, most of the gestures have really good results. Even though, the performance of the *one* class falls drastically when it comes to its recall. Apart from that, the precision of the gesture *two* is not good either. Moreover, as expressed before, this tendency was observed in the previous results too. When more photos were added to the data-set of the other gestures (*bad*, *good*, *ready* and *stop*), the performance of *one* and *two* hand gestures started to become worse. This could be related to the fact that *one* and *two* classes have two different ways of representing the same gesture as can be observed in Figure 3.9. Due to this, the ANN has at tendency to confuse these gestures with the others since there are less images of each type of *one* and *two* hand gestures. Otherwise, the fact that the identification window is bigger does not seem to affect the performance of the classification tool to classify the other gestures.

Last Approach: 10 *hand* Gestures Together

In this last approach, a big amount of classes will be tested in the ANN to see the effect that has adding more classes to the ANN. Apart from that, once good results are obtained, the training of the ANN will be continued. This way, the effect of adding more images to the training set once this is giving good results will be observed.

Location Tool In this last approach, 1314 images have been used as data-set. From that 1314 images, between the photos that are labelled, 1267 photos (80%) have been used to train the Location ANN, that is to say 1013 photos. The photos that are not labeled, 47 photos, do not have any *hand* gesture. As they are not labeled, they are not trained by the ANN. Hence, the search of hands by the Location tool is based in the photos with a person making a *hand* gesture. With this data-set the obtained results are expressed in Table 4.5.

Feature	Found	Train	Labelled	Recall	Precision	F-score
hand	1269	1013	1267	98.4	98.4	98.4

Table 4.5: Last approach: Results of the Location tool

As it can be observed in Table 4.5, the found photos are 1269, two more than the labeled ones. This means that in two photos where no-one appears or there is someone standing still, hands were incorrectly identified (see Figure 4.24).

Due to these two failures, precision and recall for the ANN is 98.4%, what leads to an F-score of 98.4%. Even the small mistake, the results are really good since all the hands were found in the gesture photos. This means that none of the hand gestures were lost in the process of the Location ANN, so the Classification ANN will get all the hand gestures for its training.

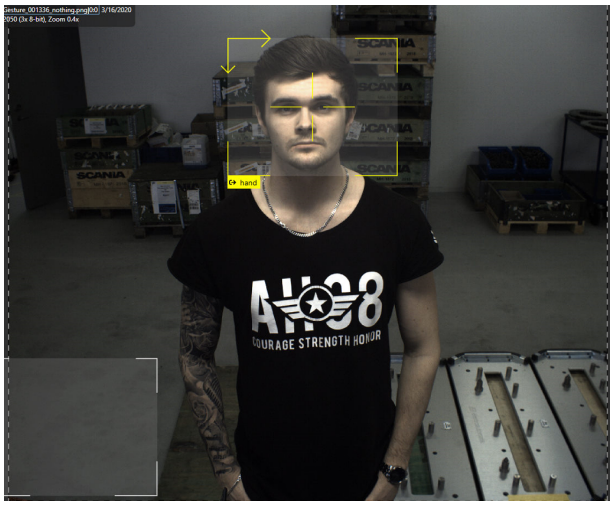


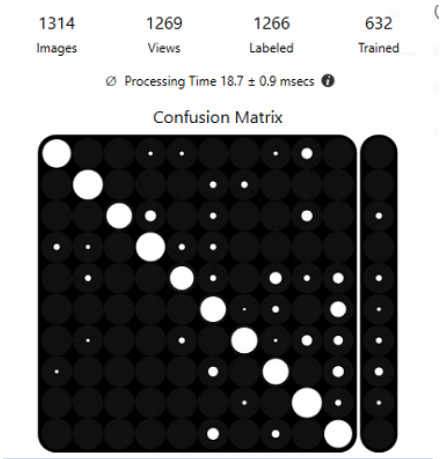
Figure 4.24: Wrongly identified hand

Classification Tool Once the Location ANN was trained, the Classification ANN has been performed. As in the previous cases, first the 50% of each gesture photos have been randomly selected to train the ANN. This way, the results in Figure 4.25, with 10 classes, were obtained.

The results obtained in Figure 4.25 show a mean recall of 76.1% and a mean precision of 81.8% over all the trained classes. All the spots out from the left up to the right down diagonal show the miss-classified images. Once, the 50% of the images is trained, miss-classified photos were added to the training set. To obtain as best results as possible, the photos added to the training set have been changed in number until three versions of ANN with pretty good results were obtained. This can be seen in Figure 4.26, Figure 4.27 and Figure 4.28. The percentages used for the training set in each version results can be observed in Table 4.6.

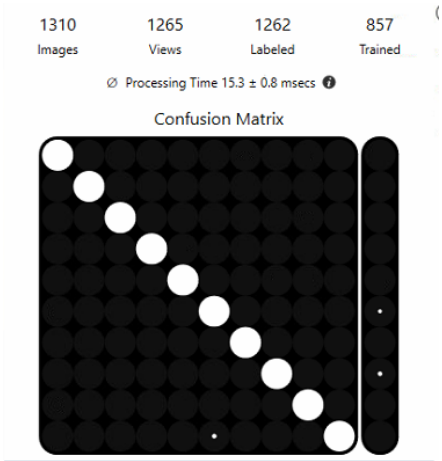
In the first obtained version of 70% trained images (see Figure 4.26) compared with the results obtained with 50% of trained images, the mean percentage of recall is improved in a 23.25% and the precision is improved in a 17.86%. Apart from this, 7 over 10 classes have a F-score of 100%, which means the recall and the precision have a result of 100% too. Hence, they have the maximum value of recall and precision. The gestures with lower values of F-score are *one*, *rock* and *two*. Even though, the F-scores of all these gestures are over the 97%, the only value that goes under this value is the precision of the *one* gesture with a value of 96.6%. Taking all this into account, and the number of images used for the training of this classification ANN [1st results in Table 4.6] can be concluded that the results obtained in Figure 4.26 are really good.

After this 1st version of the 70% trained classification ANN, more images were added to the training set obtaining the 2nd version of the ANN (see Figure 4.26). When the mean results of the 1st version are compared with the 2nd version results, a small fall in the precision and the recall can be observed. In case of the recall, a deterioration of 0.85% is reported and in case of the precision a fall of 0.80% can be seen. In this 2nd version, 3 gestures obtain a F-score of 100 over the 7 that were obtained in the 1st version. As in the previous case, all the F-scores are higher than the 97% but there is more than one gesture with values of recall and precision that are under this value. Furthermore, as commented before, more photos than in the previous version have been used for the training set and even though, worse results were obtained. Because of all these reasons, the 1st version is selected over the 2nd version.



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	109	61	118	0.0	82.5	94.0	87.9
cancel	45	21	43	0.0	90.9	87.0	88.9
give	42	24	48	4.2	66.7	100.0	80.0
good	99	51	102	0.0	86.3	91.7	88.9
middle	46	25	51	3.8	57.7	71.4	63.8
one	244	127	255	1.6	67.2	74.1	70.5
ready	137	78	157	3.8	69.6	94.8	80.3
rock	138	70	140	7.1	68.6	70.6	69.6
stop	115	49	99	2.0	92.0	69.7	79.3
two	280	126	253	0.0	79.5	65.2	71.6
	1255	632	1266	2.1	75.4	77.0	76.2
∅ (10 classes)				2.3	76.1	81.8	78.1

Figure 4.25: Last approach: Results of the Classification ANN with 50% training



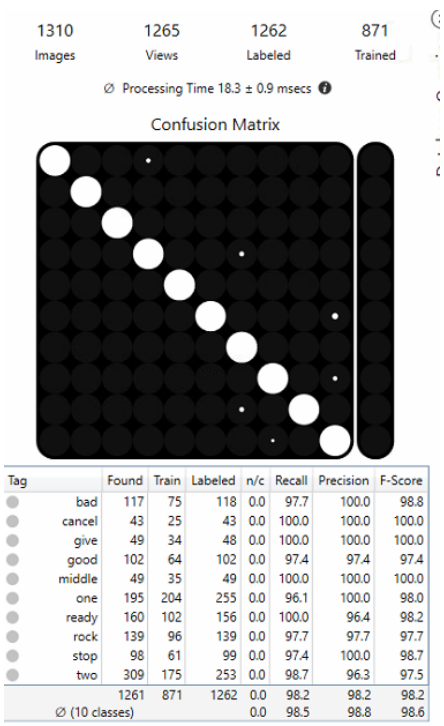
Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	125	75	118	0.0	100.0	100.0	100.0
cancel	43	25	43	0.0	100.0	100.0	100.0
give	49	34	48	0.0	100.0	100.0	100.0
good	101	64	102	0.0	100.0	100.0	100.0
middle	48	35	49	0.0	100.0	100.0	100.0
one	246	198	255	1.8	98.2	96.6	97.4
ready	154	102	156	0.0	100.0	100.0	100.0
rock	133	96	139	2.3	97.7	100.0	98.8
stop	97	59	99	0.0	100.0	100.0	100.0
two	263	169	253	0.0	97.6	100.0	98.8
	1259	857	1262	0.5	99.01	99.5	99.26
∅ (10 classes)				0.4	99.35	99.66	99.5

Figure 4.26: Last approach: Results of the Classification ANN with 70% training first version

After the 2nd version, more images were added to the training set to obtain the 3rd version (see Table 4.6). This 3rd version results will be compared with the results of the 1st one, the mean recall falls 1.95% and the mean precision falls 0.76%. As in the previous version, the F-score of three gestures of this version is 100% but the F-score goes under 97% in some hand gestures. Even more images are added to the training set, worse results than in the 1st results are obtained. This means that arrives a point when the ANN does not learn more from the newly added images. The addition of these images spite of improving the results of the ANN made them worse.

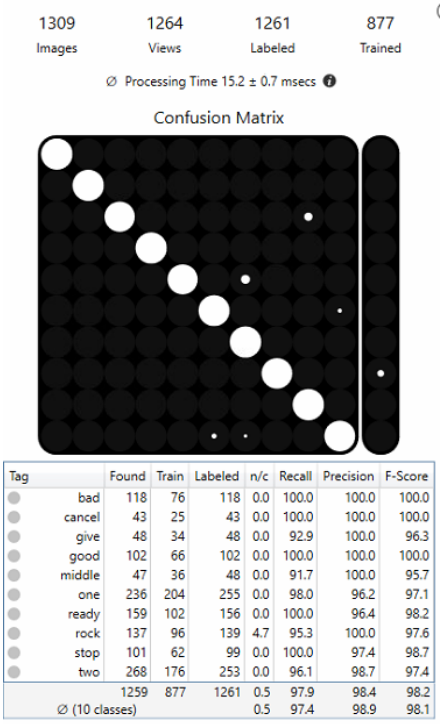
Tag	**1st results**	**2nd results**	**3rd results**
bad	63.56	63.56	64.41
cancel	58.14	58.14	58.14
give	70.83	70.83	70.83
good	62.75	62.75	64.71
middle	71.43	71.43	75
one	77.65	80.00	80.00
ready	65.38	65.38	65.38
rock	69.07	69.07	69.07
stop	59.60	61.61	62.64
two	66.80	69.17	69.57

Table 4.6: Last approach: Percentages of classification ANN



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	117	75	118	0.0	97.7	100.0	98.8
cancel	43	25	43	0.0	100.0	100.0	100.0
give	49	34	48	0.0	100.0	100.0	100.0
good	102	64	102	0.0	97.4	97.4	97.4
middle	49	35	49	0.0	100.0	100.0	100.0
one	195	204	255	0.0	96.1	100.0	98.0
ready	160	102	156	0.0	100.0	96.4	98.2
rock	139	96	139	0.0	97.7	97.7	97.7
stop	98	61	99	0.0	97.4	100.0	98.7
two	309	175	253	0.0	98.7	96.3	97.5
	1261	871	1262	0.0	98.2	98.2	98.2
Ø (10 classes)				0.0	98.5	98.8	98.6

Figure 4.27: Last approach: Results of the Classification ANN with 70% training second version



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	118	76	118	0.0	100.0	100.0	100.0
cancel	43	25	43	0.0	100.0	100.0	100.0
give	48	34	48	0.0	92.9	100.0	96.3
good	102	66	102	0.0	100.0	100.0	100.0
middle	47	36	48	0.0	91.7	100.0	95.7
one	236	204	255	0.0	98.0	96.2	97.1
ready	159	102	156	0.0	100.0	96.4	98.2
rock	137	96	139	4.7	95.3	100.0	97.6
stop	101	62	99	0.0	100.0	97.4	98.7
two	268	176	253	0.0	96.1	98.7	97.4
	1259	877	1261	0.5	97.9	98.4	98.2
Ø (10 classes)				0.5	97.4	98.9	98.1

Figure 4.28: Last approach: Results of the Classification ANN with 70% training third version

Conclusions about ANN Training

In this subsection, the conclusions obtained throughout the training of the different hand gesture ANN versions are listed,

- Number of images for each class is not closely related to the weights of the ANN. So, the same number of images in all classes are not needed to have a good performance.
- More images in a class means better results in recall and precision for that class.
- Bigger size of identification window means more information to be processed by the ANN. Due to this, more images are needed by the ANN to identify correctly the gestures in the images.
- There is a tendency of miss-classification between the *one* and *two* gestures.
- There is a tendency of miss-classification between the *two* and *rock* gestures.
- When the classification ANN is being trained, it reaches a point where the addition of images in the training set deteriorates the results of the performance of the ANN.

After applying all these obtained conclusions by the training and testing of different ANN in the last structure, the definitive ANN will be created.

4.2.3 Final Results of the ANN

In this section, the last ANN created taking into account the conclusions obtained in the training of the ANN in the last subsection will be explained and analyzed. The last ANN expressed in this section will be tested in the implementation section. To do so, all the gestures that are going to be applied in this project (*bad*, *good*, *middle*, *cancel*, *one*, *two*, *rock*, *ready* and *stop*) have been trained in the ANN in the same way as before. Hence, first the Location ANN has been trained with photos of people doing hand gestures or photos without people or photos with people standing still. After the Location tool was trained, the Classification tool has been trained first with 50% random photos for each gesture. Then, the miss-classified photos have been added to the training set until good results were obtained with a limitation of 80% of trained photos of each gesture.

Location Tool

In this final ANN, 1261 images have been used as data-set. From that 1261 images, among the 1214 photos that are labelled, the 80% have been used to train the Location ANN, that is to say 971 photos. The photos that are not labeled, 47 photos, do not include any hand gesture. As they are not labeled, they are not used to train the ANN. Hence the search of hands of the Location tool is based in the photos with a person making a hand gesture. The obtained results With this data-set are expressed in Table 4.7.

Feature	**Found**	**Train**	**Labelled**	**Recall**	**Precision**	**F-score**
hand	1214	971	1214	97.1	99.16	98.1

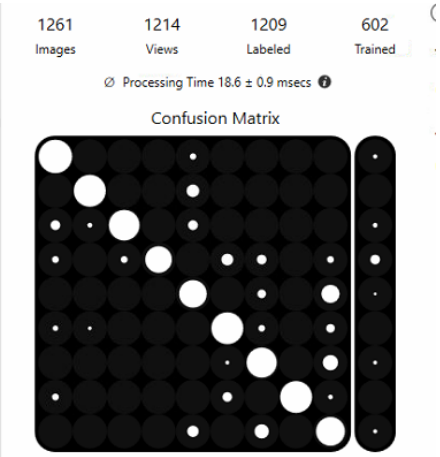
Table 4.7: Final ANN results of the Location tool

As it can be observed in Table 4.7, the number of found photos is 1214, the same number as the labelled photos. This could mean that all the hand gestures have been found correctly and that the photos without any gesture have been avoided. But, if recall and precision are observed, or F-score overall, they do not get a 100% value. So this means that some photos have not been correctly identified.

Classification Tool

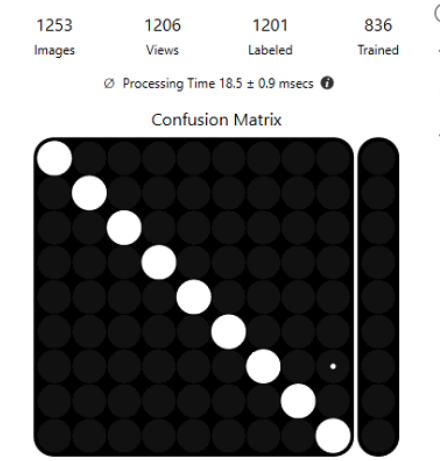
Once the Location ANN was trained, the Classification ANN was trained. As in the previous cases, first, 50% of each gesture photos have been randomly selected for training. The obtained results are shown in Figure 4.29 with the selected final 9 classes, which are *bad*, *good*, *middle*,*cancel*, *one*, *two*, *rock*,*ready* and *stop*.

When results in Figure 4.29 are compared with those in Figure 4.30, the improvement of the mean recall value which has improved 21.42% and the precision value that has improved 15.03%, can be observed. All F-scores are over the 98% of performance, which is a really good result. The lowest value is the recall for the *rock* gesture, which is 97.5%, followed by precision for the *two* gesture, which is 98.5%. This is due to the miss-classification of *rock* gesture with *two* gesture as it can be observed in the confusion matrix in Figure 4.30. Although the results have improved, in *middle* and *one* gestures



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	125	59	118	1.7	94.9	86.2	90.3
cancel	41	20	42	0.0	86.4	90.5	88.4
good	91	51	100	2.0	79.6	97.5	87.6
middle	39	24	49	8.0	60.0	100.0	75.0
one	231	126	253	0.8	64.6	78.1	70.7
ready	154	78	156	0.0	85.9	89.3	87.6
rock	158	69	139	1.4	77.1	60.7	67.9
stop	93	49	99	0.0	86.0	100.0	92.5
two	273	126	253	1.6	70.1	61.0	65.2
	1205	602	1209	1.3	76.4	77.5	76.9
∅ (9 classes)				1.7	78.3	84.8	80.6

Figure 4.29: Results of the final hand gesture Classification ANN with 50% training



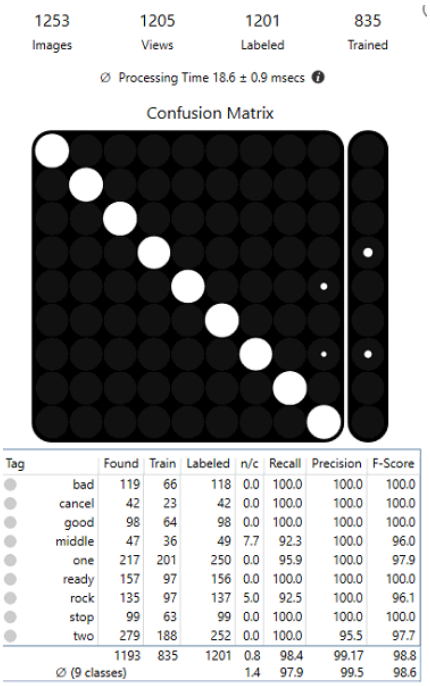
Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	118	66	118	0.0	100.0	100.0	100.0
cancel	42	23	42	0.0	100.0	100.0	100.0
good	98	64	98	0.0	100.0	100.0	100.0
middle	49	40	49	0.0	100.0	100.0	100.0
one	240	201	250	0.0	100.0	100.0	100.0
ready	158	95	156	0.0	100.0	100.0	100.0
rock	136	97	137	0.0	97.5	100.0	98.7
stop	99	63	99	0.0	100.0	100.0	100.0
two	264	187	252	0.0	100.0	98.5	99.24
	1204	836	1201	0.0	99.73	99.73	99.73
∅ (9 classes)				0.0	99.72	99.83	99.77

Figure 4.30: Results of the final hand gesture Classification ANN with 70% training 1st version

the percentage of needed images exceeds the 80%. In the case of the *middle* hand gesture this number is exceeded by a 1.63% and in the case of *one* gesture 0.4% (see Table 4.8). Even though these exceeding percentages are not big, the ANN will be newly trained with less images of the *middle* hand gesture. Before the performance of the training of this ANN a deterioration of the results is expected, since the *middle* gesture will have less images to work with.

When this second version of the 70% trained classification ANN is run (see Figure 4.31) and compared with the previous version (Figure 4.30), as expected, a fall of the mean recall and precision values can be observed. The recall falls a 1.82% and the precision a 0.33% compared with the previous version. Apart from that, the lowest F-score value corresponds to *middle* hand gesture, which was predictable since the number of images used to train this gesture has been reduced in four. In general, all values of recall and precision have dropped compared with the previous version. An interesting result is of the *rock* gesture, since the amount of trained images of this gesture had an increase which lead to a fall of 5% in its recall. This means that the reduction of images for the training of the *middle* gesture is indirectly affecting especially the results of *rock* gesture.

When the results of both ANN (Figure 4.30 and Figure 4.31) are observed, even though in the 1st version (see Figure 4.30) the percentage of used images for the training of the ANN exceeds the 80%, better results than in the previous version are obtained. So, the ANN expressed in the Figure 4.30 will be selected as the final ANN and tested in the implementation section.



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	119	66	118	0.0	100.0	100.0	100.0
cancel	42	23	42	0.0	100.0	100.0	100.0
good	98	64	98	0.0	100.0	100.0	100.0
middle	47	36	49	7.7	92.3	100.0	96.0
one	217	201	250	0.0	95.9	100.0	97.9
ready	157	97	156	0.0	100.0	100.0	100.0
rock	135	97	137	5.0	92.5	100.0	96.1
stop	99	63	99	0.0	100.0	100.0	100.0
two	279	188	252	0.0	100.0	95.5	97.7
	1193	835	1201	0.8	98.4	99.17	98.8
∅ (9 classes)				1.4	97.9	99.5	98.6

Figure 4.31: Results of the final hand gesture Classification ANN with 70% training 2nd version

Tag	50% results	70% 1st results	70% 2nd results
bad	50.00	55.93	55.93
cancel	47.62	54.76	54.76
good	51.00	65.31	65.31
middle	48.98	81.63	73.50
one	49.80	80.40	80.40
ready	50.00	60.90	62.18
rock	49.64	70.80	70.80
stop	49.50	63.64	63.64
two	49.80	74.21	74.60

Table 4.8: Final Accuracy percentages for Classification ANN

4.3 ANN for Task 1

In this section, the development and analysis of the created ANN for the Task 1 is reported.

4.3.1 Structure Creation of the ANN

The structure of the ANN for this task is composed by two tools. Firstly, the Location tool is used for the same reason as in the hand gesture case, to crop the industrial piece from the remaining part of the image (see Figure 3.14 and Figure 3.15). Once the Location tool is applied and the industrial part is separated from the rest of the image, the Analyze tool is used. This tool first searches for small details in the image that allows to classify the image as *bad* (images with a marking in the piece) or *good* (images without any marking in the piece). In this case, the small details are the marks done in the industrial part, which

simulate errors in the piece. So the final structure of the ANN for the Task 1 is the combination of Location and Analyze tools as it can be observed in Figure 4.32.

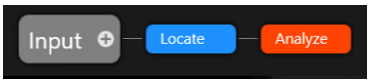


Figure 4.32: ANN structure for the Task 1

4.3.2 Training of the ANN

In this part, the training of the whole ANN is explained. Firstly, the process and obtained results of the Location tool will be expressed. Next, the different approaches made by the Analyze tool will be listed.

Location Tool

As expressed before, the Location tool is used to find and separate the part of the image that is interesting for the ANN, in this case the industrial piece. To do so, the images listed in Section 3.5.2 have been uploaded to the ANN. Then, as in the hand gesture ANN, the region of interest has been decided as all the image since the camera will take that region. Next, a model was created since the industrial piece will always be placed in the same position within the image. The model is an area within the target has to be placed. This allows to easily locate the wanted part of the image since the marker should be inside the created model. After the model is applied, the industrial piece of each image has been pointed with the marker of the Location tool (see Figure 4.33).

Figure 4.33: Task 1 Location tool application image

The first square visible in the image, the biggest one, is the applied model to the image. This model has a number on it that represents if a marker has been found inside the selected area. If the number is 0/1 means that there was not a marker inside the model. Whereas, if the number is 1/1 means that a marker has been found inside the model. The second square (the one inside the model square) is the actual marker that finds the piece in the image, which has the size of the defined feature size.

As in this case a model was applied, in the results two different tables were obtained. On the one

hand, the table representing the actual results of the performance of the ANN. On the other hand, the results that represent the performance of the model.

Feature	Found	Train	Labelled	Recall	Precision	F-score
Task1	80	40	80	100	100	100

Table 4.9: Markers results of the Task 1 ANN

Model	Found	Train	Labelled	Recall	Precision	F-score
Model 1	80	40	80	100	100	100

Table 4.10: Models results of the Task 1 ANN

As it can be observed in the two tables (see Table 4.9 and Table 4.10) with a 50% of images used for the training of the ANN and the other 50% for test, a 100% of recall and precision are obtained, which is the best case scenario with not failures at all.

Analyze Tool

As explained in the beginning of this section, the Analyze tool allows to analyze the image given to the tool and find differences between the images labelled as good and bad. The tool gives the option to train the ANN as what are called "supervised" or "unsupervised" methods. Even though, following the description of what is the difference between supervised and unsupervised methods in machine learning [46] is the existence of labels in the training set. The unsupervised method suggests pattern recognition without the implication of a target attribute.

In the ViDi software, when the "unsupervised" method is used, the images are labeled as good or bad but the details that make the difference between them are not marked. This way of training the ANN is incorrectly called unsupervised in this software since the images are labelled, hence it is a supervised method. When the "supervised" method is used, apart from labelling the images as good or bad, the scratches or markings in the target are manually marked.

First Approach: Supervised Method Without Markings First the supervised method without marking the errors in the industrial piece has been used for the training of the ANN, which in the ViDi software is called as "unsupervised" method. In this case, the images are labelled as *good* or *bad*. On the one hand, if the industrial piece in the image has markings, then the image will be labelled as *bad*. Whereas if the industrial part does not have any markings the image will be labelled as *good*. Once all the images were labelled, 50% of the images of the database were used for the training of the ANN and the remaining 50% of the image database were used for the testing set. As it is observable in Figure 4.34, Figure 4.35 and Figure 4.36, there are three possible results. When the image is correctly identified as good piece (see Figure 4.34), when the image is correctly identified as bad piece (see Figure 4.35) and when the piece could not be identified neither as good or bad (see Figure 4.36).

In the Figure 4.34, in the right upper corner, a green tag can be seen which means the image was labelled as *good*. Apart from that, the square surrounding the image is green too, this square represents

Figure 4.34: 1 st approach: Correct piece after the Analyze ANN

the identification tag that the ANN gave to the image.

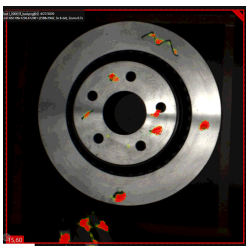


Figure 4.35: 1 st approach: Incorrect piece after the Analyze ANN

In the Figure 4.35 image, the red marks in the middle of the image are the found errors in the photo, depending in the level of the found markings the image is identified as *good* or *bad*. As in the previous image, the label given to the photo is represented by the colour of the line placed in the upper right corner of the photo, which in this case is red so is a bad piece. Even though, if the image is observed, there are some error markings out the industrial piece, which means the ANN found incorrectly errors in the image.

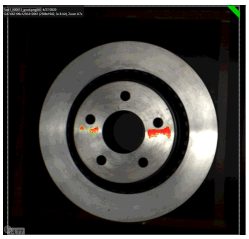


Figure 4.36: 1st approach: Unidentified piece after the Analyze ANN

The Figure 4.36 shows an error made by the ANN when it comes to the search of mistakes in the industrial piece since the piece is a good part and does not have errors. As the square surrounding the image is gray, the photo has not been identified as neither good or bad. The results obtained from this ANN can be observed in Figure 4.37 and Figure 4.38.

The Figure 4.37 and Figure 4.38 show the same results in different graphic representations. In the left image (see Figure 4.37), the TP, TN, FP and FN are easily identified. The true positives, the images identified correctly as good, are the 18 images located in the green slope. The true negatives, the 52 images identified correctly as bad, are the images located in the red slope. The false positives, the images identified incorrectly as good, are the 4 images positioned inside the gap between the two vertical lines in red colour. To finish, the false negatives, the images identified incorrectly as bad, are the 3 images positioned inside the gap in green colour. The same results are represented in Figure 4.38 in form of ROC curve and AUC value, this has a value of 0.992% which is a really good result with a 99.2% of

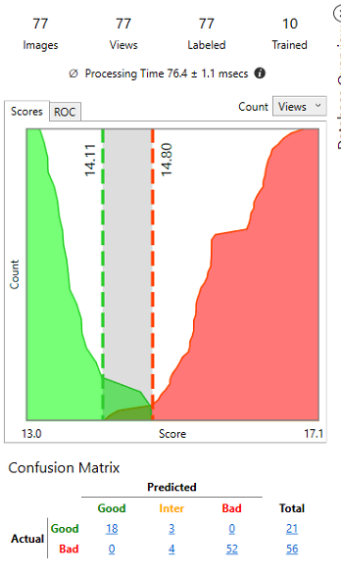


Figure 4.37: 1 st approach: Results of Analyze tool ANN in confusion matrix

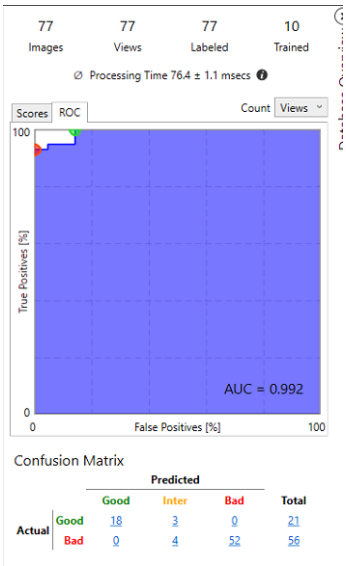


Figure 4.38: 1 st approach: Results of Analyze tool ANN in ROC curve

performance of the ANN. Even though, the total 7 miss-classified images, the sum of FP and FN, are wanted to be classified correctly. To do so, the other mode of the software is tested, what is called as supervised method in the software, this way the AUC of the ANN will be higher which means a better performance of the ANN will be obtained.

Second Approach: Supervised Method With Markings In this second approach, the supervised method of the software with a manual marking of the errors in the pieces will be tested. The feature size of the ANN is of 40 pixels and the colour sample is of 4 channels which are CMYK (cyan, magenta, yellow, and key (black)). Apart from that, as it can be observed in Figure 4.39 the errors in the bad pieces have been manually marked, this is easily identifiable if this figure is compared with Figure 4.35. This way, mistakes in the error (scratches and markings) search of the industrial piece are avoided. As it is observable in Figure 4.39 the markings are pretty thick and not really concise. With this settings, the results listed in the Figure 4.40 and Figure 4.41 have been obtained.

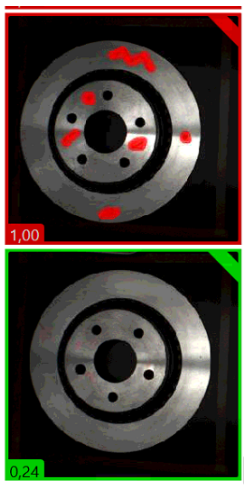


Figure 4.39: 2nd approach: Marking of the errors in the industrial piece

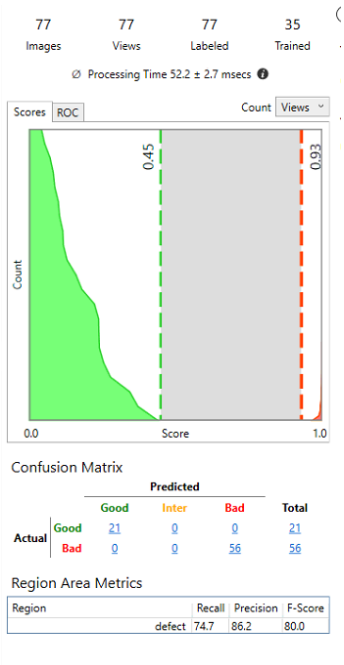


Figure 4.40: 2nd approach: Results of Analyze tool ANN in confusion matrix

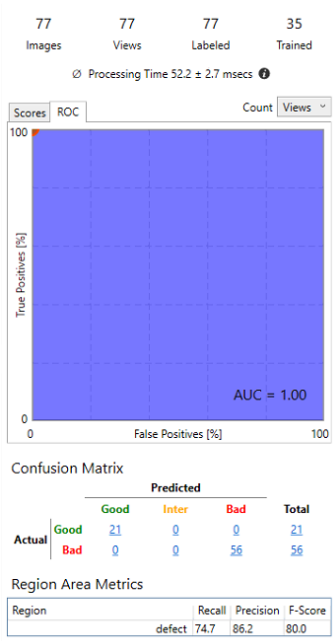


Figure 4.41: 2nd approach: Results of Analyze tool ANN in ROC curve

In this results, apart from the information given by the ROC curve and the Scores graphic, the region area metrics outcome is given. This last results give the information about the performance of the size of the markings made in the images. To obtain the results represented in these images, 45% of the database has been used for training purposes and the remaining 55% as testing set. As predicted before, with the manual markings of the errors in the industrial piece better results than in the previous approach are obtained. Hence, the AUC has increased to a 100% of performance which is observable in Figure 4.41 as well as the lack of miss-classified images in Figure 4.40. This is easily observable in Figure 4.40 in the empty gap between the vertical lines. This means there are not FP or FN in the results of the ANN which is translated in a perfect performance in the prediction of the ANN. Apart from that, the region area recall is of 74.7% and the precision is of 86.2%. Since the performance of the ANN is perfect with a AUC of 1 and a really low percentage of images is used for the training set, the only thing to be improved is the precision and the recall of the marking size. To improve these results, the only parameter to be changed is the feature size, because of that reason different feature sizes have been tested to see how the results of the region area metrics are affected.

Approach	Feature size	Recall	Precision	F-score
3	40 pixels	93.4	70.4	80.3
4	19.9 pixels	69.3	96.7	80.7
5	30 pixels	73.1	89.1	80.3
6	33 pixels	84.6	82.2	83.4

Table 4.11: Different feature size Task 2 Classification tool table

From these tests (see Table 4.11) the best possible and balanced results for this ANN can be concluded, these are obtained when the feature size is near 33 pixels.

4.3.3 Final results

In this last approach, the supervised method with hand made markings has been used. In this case, the markings are more precise as it can be seen in Figure 4.42 since precision and recall of region area matrix have improved compared with the previous approach (see Figure 4.39. Apart from that, the feature size has been changed to 32.5 pixels and 53% of the images in the database has been used for the training of the ANN, leaving the remaining 47% for the testing set.

Figure 4.42: Final approach: Marking of the errors in the industrial part

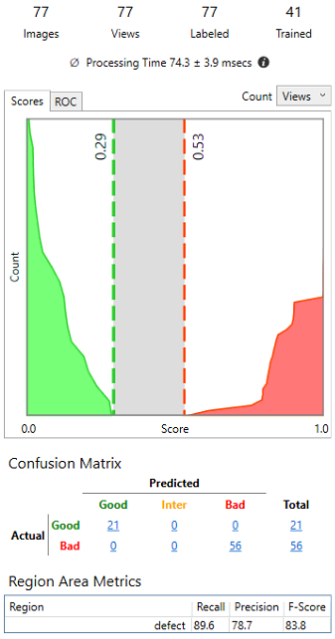


Figure 4.43: Final approach: Results of Analyze tool ANN in confusion matrix

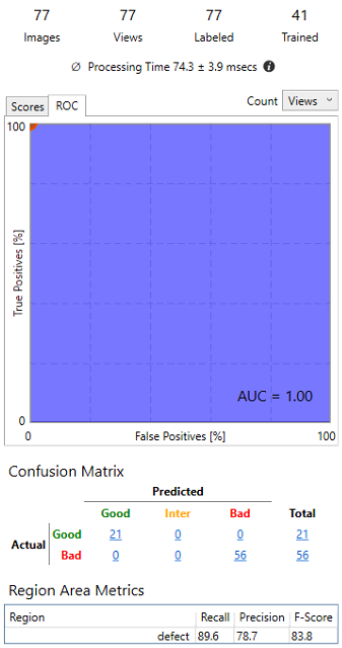


Figure 4.44: Final approach: Results of Analyze tool ANN in ROC curve

The results obtained in this approach can be observed in Figure 4.43 and Figure 4.44. As in the previous approach, the AUC remains in a 100%, which means a perfect performance of the ANN and non confusion between the two different labels. This can be observed in the empty gap between the two vertical lines of the scores graphic too. So, the obtained improvement by the change of the precision of the markings in the industrial piece is the performance of the region area metrics, which has increased in a 3.8% in the F-score. Apart from that, the area marked by the vertical lines is more centered in the graphic and the TP (green slope) and TN (red slope) are more identifiable.

4.4 Task 2: Muffin Chihuahua Dice Game ANN

In this section, how the structure and the training of the chihuahua muffin dice ANN has been performed is explained and analysed. As reported before in Section 3.3.2, the main objective of this task is to get 5 chihuahuas, what is called to get a Yattzy, to win the dice game.

4.4.1 Structure Creation of the ANN

The structure for this ANN has two parts, the location tool and the Classification tool. Hence, the structure of this ANN is the same as the structure of the hand gesture ANN (see Figure 4.13). As in the previous cases, the Location tool is used to crop the parts of the image that are important for the training of the ANN. In this case, the important parts of the taken image are the dices with muffin and chihuahua photos. When the Location tool is applied, the box which contains the dices is found within the image and cut out from the rest of the photo. After the Location tool is applied and the wanted parts of the image are obtained, the Classification tool is applied to classify the dices photos as muffins or chihuahuas. This way, apart from finding the chihuahuas and muffins in the image, the amount of them can be counted.

4.4.2 Training of the ANN

As expressed in the structure creation of the ANN, the ANN has two parts, so in this subsection the different tried parameters will be expressed with the used two tools (Location and Classification tools). For the training of this ANN, 59 images of the muffin/chihuahua dices in different positions and with different images of chihuahuas and muffins have been taken.

Location Tool

As explained before, the ANN first part is the location tool with the aim of searching for the dice in the image. This can be observed in Figure 3.16 where the location squares are placed in the dices. For the training of this ANN, over the 59 images that were taken to create the database for this task, 30 were used for the training set and the remaining 29 for the testing set. This means that the 50.85% of the images of the database were used for the training of the ANN. As in each taken image there are 5 dices, the total amount of labeled dices are 295 and as 50% of them are used for the training set this leads 150 labelled dices for the training set and 145 for the testing set. With this set parameters, the results observable in Figure 4.12 were obtained.

Feature	Found	Train	Labelled	Recall	Precision	F-score
Dice	295	150	295	100	100	100

Table 4.12: Location results for the Task 2

As it can be observed in Figure 4.12, recall and precision obtained by this ANN is 100% which can

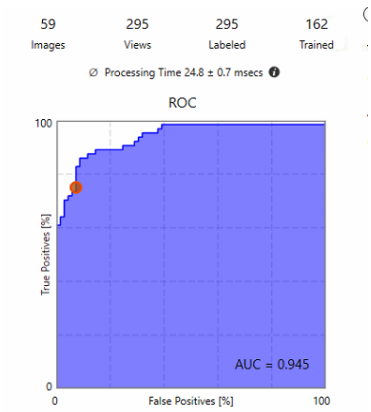
not be improved more since it is the highest score that can be obtained. Hence, this location ANN will be the final one.

Classification tool

After applying the Location tool, the Classification tool has been applied to classify the cropped images between muffins and chihuahuas. This can be observed in Figure 4.45. The ANN has 295 images like the one pointed in Figure 4.45 and the number of muffins and chihuahua images is not the same.

Figure 4.45: Cropped dice image by the Location tool

First Approach: Classification Tool with Dices with a Feature Size of 98.8 Pixels In this 1st approach of the Classification tool, the used database was of 295 images of chihuahuas and muffins, 162 images were used as training set and the remaining 133 were used as testing set. This means that 55% of the total number of images have been used for the training set and 45% for the testing set. Moreover, the used feature size is 98.8 pixels with a luminance of 30%, a contrast of 20% and a colour sample of 3 channels (RGB). With these parameters, the results printed in Figure 4.46 were obtained.



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
dog	131	78	142	0.0	75.0	90.6	82.1
muffin	164	84	153	0.0	92.8	80.0	85.9
	295	162	295	0.0	84.2	84.2	84.2
Ø (2 classes)				0.0	83.9	85.3	84.0

Figure 4.46: 1st approach: ROC curve of the Classification tool in Task 2

The AUC obtained from this ANN is 94.5%, which is pretty good but can be improved. The precision result of the dog tag (referred to the chihuahua dices) is 90.6% which is acceptable, but recall falls to 75% which is improvable. The muffin tag has better results with a recall of 92.8% and a precision of

80%. This leads to a mean recall value of 83.9% and a precision of 85.3%. These values are wanted to be improved as much as possible, to do so the feature size will be changed until as best results as possible are obtained with the actual database.

The different tested feature sizes and their correspondent mean precision, recall and F-score results are listed in Table 4.13.

Approach	Feature size	Mean Recall	Mean Precision	Mean F-score
2	18.1 pixels	96.9	97.3	97.0
3	25.4 pixels	96.9	97.3	97.0
4	34.1 pixels	94.5	95.4	94.7
5	10.4 pixels	96.9	97.3	97.'

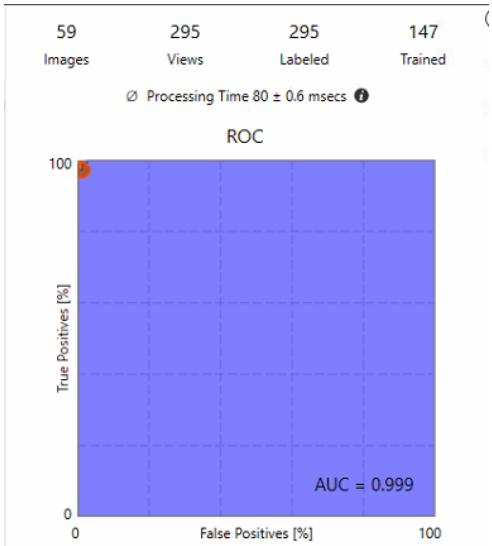
Table 4.13: Different feature size Task 2 Classification tool table

The different tests related to the feature size, do not give a lot of information about the tendency of the ANN when the feature size is changed. Even though, it is clear that a bigger feature size gives worse results than a smaller one. Because of that reason, the ANN will be trained with the smallest feature size as possible, that is 6.58 pixels.

4.4.3 Final Results

The final ANN for the 2nd Task is the combination of the already explained Location ANN in Section 4.4.2 and the Classification ANN explained in this subsection which can be observed in Figure 4.47. In this final Classification ANN, the feature size has been settled to 6.58 pixels and the tool parameters have been adjusted to the same parameters as in the 1st approach (contrast, colour channel etc.). The obtained results by this specifications are observable in Figure 4.47.

In this case, the mean recall has improved 1% and the mean precision has improved 0.7% which means that the mean F-score has improved 1%, compared with the last approach made in the previous subsection, the 5th approach. This last approach has a F-score of 98%. Apart from that, the AUC of the ROC curve gets a performance of 99.9% which is really near to 100%, the perfect score, that is a really good result.



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
dog	141	71	142	0.0	97.2	98.6	97.9
muffin	154	76	153	0.0	98.7	97.4	98.1
	295	147	295	0.0	98.0	98.0	98.0
Ø (2 classes)				0.0	97.9	98.0	98.0

Figure 4.47: Final approach: ROC curve of the Classification tool in Task 2

Chapter 5

Simulation Phase of the Robot

In this section, the simulation of the YuMi robot will be explained. Before the implementation of the whole project in the real robot and the real environment, a simulation of the movements, TCP/IP connection (see Section 2.1.6) and synchronisation of the arms of the robot (Section 2.2.4) has been performed.

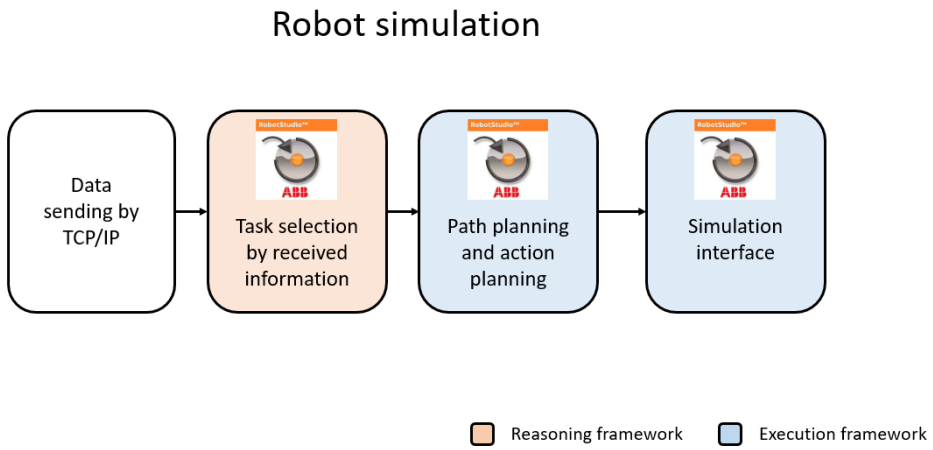


Figure 5.1: Workflow of the simulation of the robot

The YuMi robot (Section 2.1.1) will make two different types of movements which will depend in the action to be performed. The movements can be synchronized, when both arms do the same gestures but in mirror effect, or individual in case each robot arm has to perform a different movement.

The programming of the YuMi robot has been coded in RAPID language by the use of RobotStudio program (Section 2.1.3). As the arms of the YuMi robot are designed as individual robot arms with the same base, each robot arm needs a program with a main module and a data module. Due to this, two programs with the names `T_ROB_R` and `T_ROB_L` have been created, the first one refers to the right robot arm and the second one to the left arm. Apart from that, another program has been created for the client server function of the TCP/IP connection called `TCP_IP_`. This three programs work in parallel so that none of the programs interfere the other ones. To watch the results of the previously commented programs in a simulation in RobotStudio, please visit `https://youtu.be/iJIk3bHi0-4`.

5.1 Robot Arm Programs

As expressed before, each robot arm has its own program which is composed by a data module and a main module. The main objective of the data module is to define the global variables shared with the other modules. Whereas the main module is composed by the functions and instruction for the correct functionality of the robot. As the name itself says, the main module is the main part of the program. Both programs follow the same structure that can be observed in Figure 5.2.

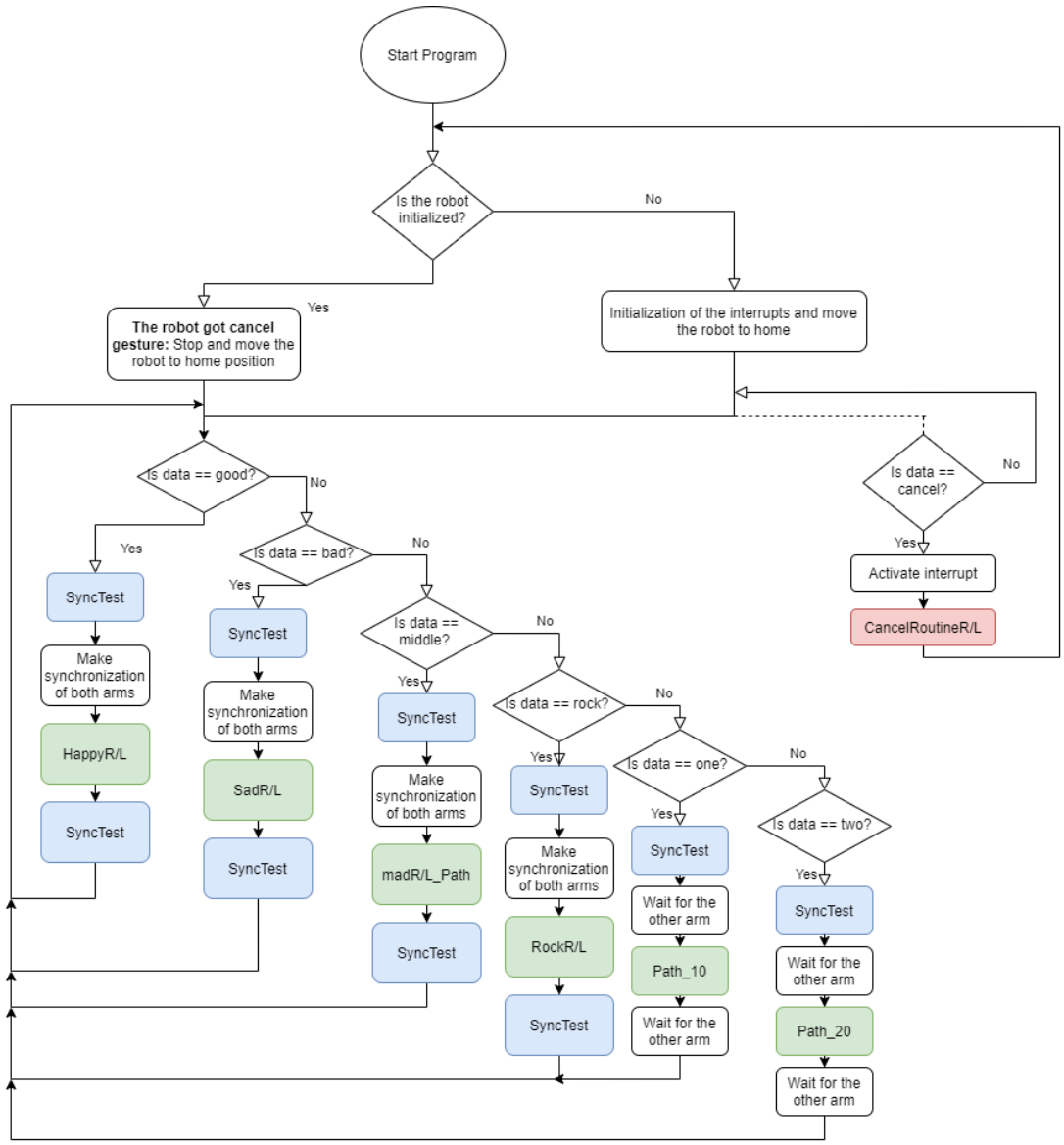


Figure 5.2: Flow chart of the robot arms

As it can be observed in the flow chart of the robotic arms, some actions are colored. The blue action is a function, the green ones are paths followed by the robot and last but not least, the red is the routine that is run when the interrupt of cancel gesture is given.

SyncTest Function

The *SyncTest* functions main job is to check whether the synchronization of both arms is active and in that case to brake the synchronization between them. This way, the synchronization state of both arms is ensured and clear, since sometimes a previous synchronization was stuck and did not allow to continue with the execution of the program. This function, as it can be observed in the Figure 5.2, is used many times in both robot arm programs. As expressed before, this function is coded in both main modules of both robot arms.

Robot Paths

When it comes to the actions remarked in green, the robot paths, are functions which contain defined movements with specific points that were previously established. Two different types of movements have been used, which are *MoveJ* and *MoveL*. On the one hand, *MoveJ* enables the robot to move the joints freely from the previous position to the defined new position. On the other hand, *MoveL* creates a linear path from the previous position to the new position. Apart from the type of movement, the speed, tool and error zone are defined. The error zone is the radius where the robot can move out from the established path. Moreover, as it can be seen in the Figure 5.2 each path function has *R/L* written, this depends in the actual selection of the program. If the right arm program is selected, the function will be named *HappyR*, whereas if the program is of the left arm, the function will be named *HappyL*.

One important point to be noted is that this flow charts correspond to the simulation of the robot. Hence, the *path_10* and *path_20* are invented movements to check the asynchronized movements of the robot arms. Furthermore, all the other paths (*Happy*, *Sad*, *mad* and *rock*) are synchronized movements of the robot used to express a feeling, which will be used in the implementation phase.

Cancel Routine

The cancel routine, the action in red, is executed when the interruption is given. The interruption is activated when the *cancel* gesture is provided. When the interrupt is given, robot stops immediately whatever it is performing in that moment and jumps to the routine. When this routine is run, if there is a synchronization between the two arms, this is broken. When arms are not synchronized, the cycle is finished by the use of previously created function called *ExitCycle*. This way, the program jumps to the beginning of the program and moves the robot to the home position.

Synchronization of both Robot Arms

Another important point of this program is the synchronization of both robotic arms, `T_ROB_R` and `T_ROB_L`. To do the synchronization and the desynchronization of both robotic arms, three predefined functions have been used.

- **WaitSyncTask** The program waits in the point this function is written, until the other program

arrives to the same *WaitSyncTask* of its code. This way, both programs (`T_ROB_R` and `T_ROB_L`) are in the same point of the code.

- **SyncMoveOff** This function allows to desynchronize the robot arms that are pointed.
- **SyncMoveOn** This function allows to synchronize the robot arms that are pointed.

5.2 `TCP_IP_` Program

To create the TCP/IP connection, server-client sockets have been programmed in the `TCP_IP_` programs main module. As in the previous case, this program is composed by a data module and a main module. The flow chart of this program can be observed in Figure 5.3.

As in the previous program, the function of this program has been remarked in blue. There is only one function which is called *StartErrorHandling*. This function, is executed after the robot is restarted after a stop. Sometimes, the start predefined RAPID function does not work correctly, when that happens this function ensures that the robot starts to move.

The *setup* function, that is remarked in orange, runs only once when the robot is started. This function is used to initialize variables which need specific value in the start of the program.

Apart from these two functions, there is an error handler that is a special type of program, which is not a function, routine or interruption. This error handler is programmed in the final part of the main program and runs when there is an error in the creation or connection of the socket. When there is an error, this error handler fixes the error by repeating the process of the creation of the sockets and making the connection again.

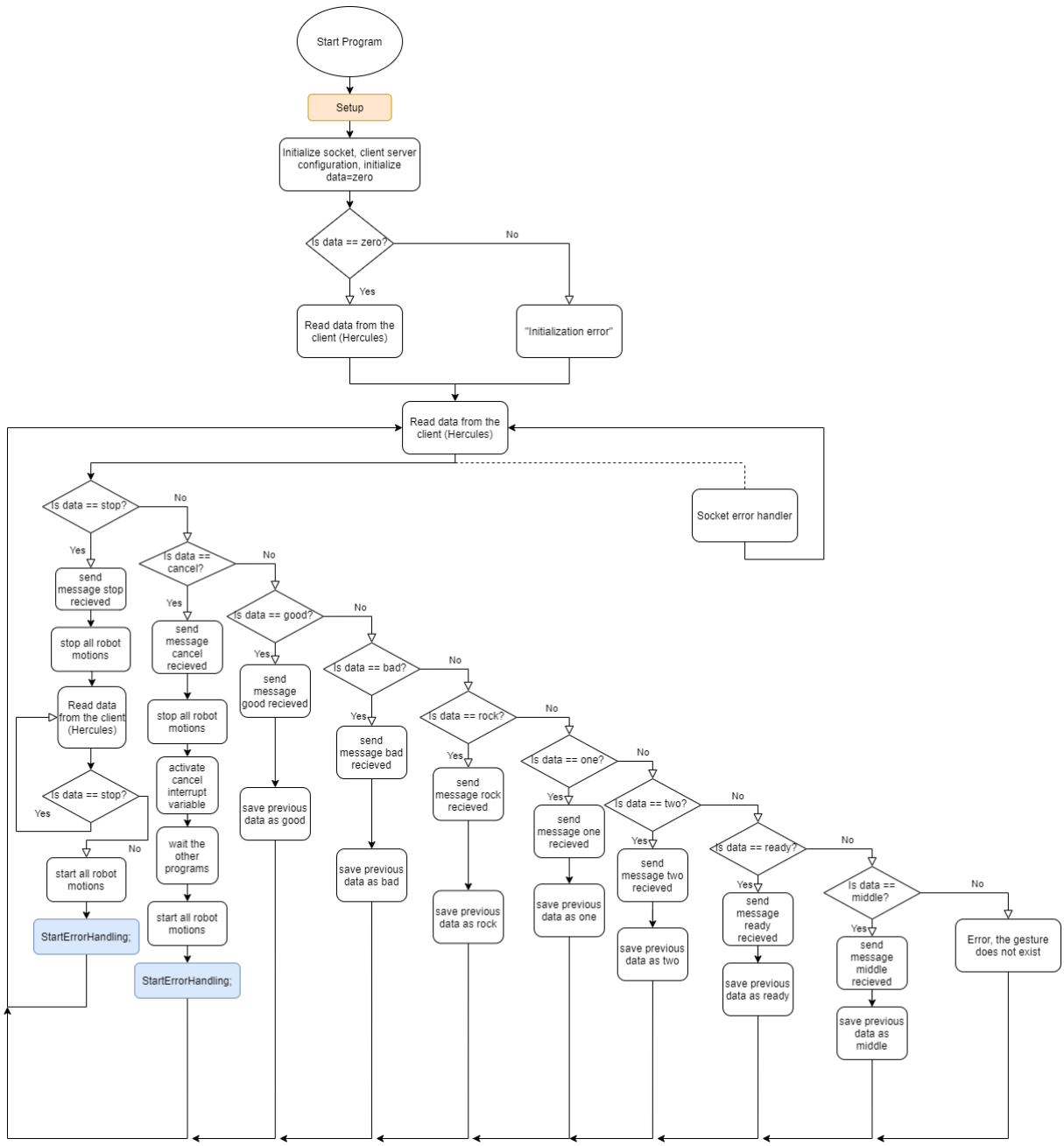


Figure 5.3: Flow chart of the TCP_IP program

Chapter 6

The Implementation Phase

In this phase, all the needed steps to get the whole implementation of the project are explained. To do so, each framework implementation will be explained and made separately to ensure the correct functionality of each framework. Hence, the vision, recognition and execution frameworks are reported. The workflow of the entire implementation of the project can be observed in the Figure 6.1.

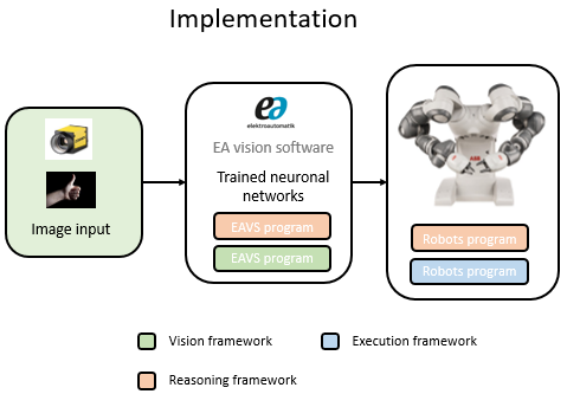


Figure 6.1: Workflow of the final implementation phase

6.1 The Visual Perception Framework

Once the ANN's are trained, their performance in real environment has been tested. To do so, images for the three ANN's have been taken, that is to say images of hand gestures and the two tasks. This lead with a new database of 114 for hand gestures, 74 photos of the industrial piece and 26 photos of the box with the dices (see Table 6.1).

Hand gesture	Industrial piece	Dices
114	74	26

Table 6.1: Number of images in the image database for the visual perception framework testing

These images have been uploaded to the ViDi vision pro Cognex software to the already trained ANN's. This software has a "production mode" to test the already trained ANN with the new images (see Figure 6.2). This way, the efficiency of the ANN with the new images is tested.

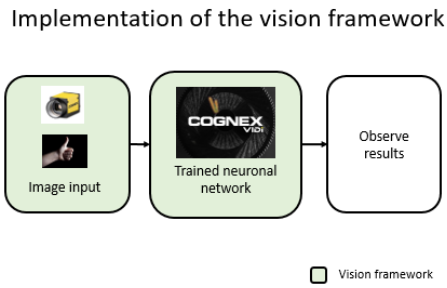


Figure 6.2: Implementation of the vision framework

6.1.1 Trained ANN test

In this subsection, the obtained results after the testing of the images listed in Table 6.1 are displayed.

Hand Gesture ANN

In this case, 114 new images have been taken to test the hand gesture ANN. The change of the building where the images for the training of the hand gesture ANN must be remarked. This means that the ambient light and the position of the cameras is different for the different image data-sets. That is to say, the data-set used for the training of the ANN and the data-set taken for this implementation test.

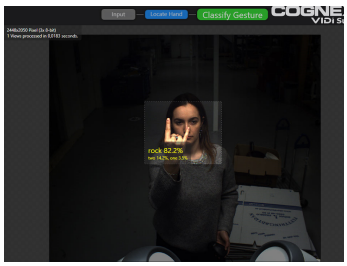


Figure 6.3: Rock Gesture test correctly classified

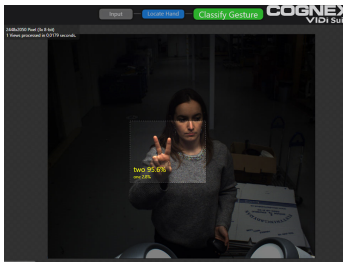


Figure 6.4: Two gesture test correctly classified

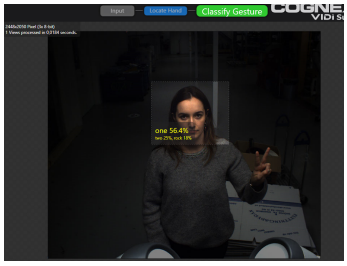


Figure 6.5: Face incorrectly taken as hand

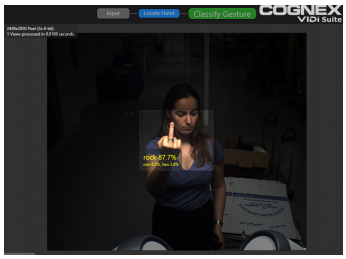


Figure 6.6: One gesture test incorrectly classified

As it can be observed in the previous images, some of the hand gestures were incorrectly classified (see

Figure 6.5 and Figure 6.6) and some other correctly (see Figure 6.3 and Figure 6.4). Taken into account the amount of miss-classified images, the performance of this ANN is not acceptable. This deterioration of the performance of the ANN is due to the change of the placement of the structure, the environment light and position of the cameras. Hence, the newly taken images, the ones taken for the test of the implementation of the hand gesture ANN, will be added to the final ANN for its training.

Apart from adding the new images to the ANN, western Europe *one* and *two* gestures have been erased (see Figure 3.9). This is due to the fact that the project is not going to be exposed in a technological fair, spite it is going to be placed in the company where most of the workers are Swedish. So, for them, the Swedish number signs were easier and more comfortable to make. Moreover, the possibility to select the same task with two different gestures was confusing for some of the users. The obtained results of the final ANN, with the specifications commented in this paragraph can be observed in Figure 6.2 and Figure 6.7.

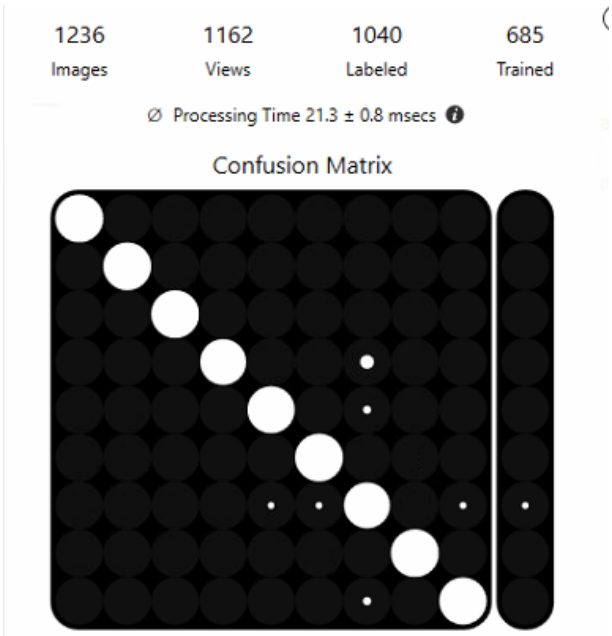
Feature	**Found**	**Train**	**Labelled**	**Recall**	**Precision**	**F-score**
hand	1162	810	1165	96.9	98.9	97.9

Table 6.2: Locate results: Final hand gesture ANN after implementation

Once the western Europe *one* and *two* gesture images were erased, the number of obtained images is of 1236. From that 1236 images, 1165 had hand gestures and the remaining 71 images have people standing still or passing in front of the camera, without making any hand gesture. If the results in Figure 6.2 are observed, of the 1165 labelled images 1162 were found. This means that the images where people is not making hand gestures are not taken into account and three images with hand gestures have not been found by the ANN. Nevertheless, an F-score of 97.9% has been obtained with a recall of 96.9% and precision of 98.9% which are really good results. Once the locate tool was trained, the classification tool has been trained and the results represented in Figure 6.7 were obtained.

- **One:** 105/140 →75% trained.
- **Two:** 98/131 →74.8% trained.
- **Stop:** 56/99 →56.56% trained.
- **Bad:** 71/122 →58.2% trained.
- **Ready:** 100/166 →60.24% trained.
- **Good:** 74/106 →69.81% trained.
- **Cancel:** 44/79 →55.7% trained.
- **Rock:** 97/145 →66.9% trained.
- **Middle:** 40/52 →76.92% trained.

With a feature size of 37.2 pixels, 1162 hand gesture images and the trained percentages represented in the previous points the results described in Figure 6.7 have been obtained. As it is observable, all the recall and precision values of the different tags are higher than 91%, which means all the tags are similarly well classified. The *bad*, *cancel*, *good* and *stop* gestures have 100% of recall and precision which



Tag	Found	Train	Labeled	n/c	Recall	Precision	F-Score
bad	127	71	122	0.0	100.0	100.0	100.0
cancel	115	44	79	0.0	100.0	100.0	100.0
good	113	74	106	0.0	100.0	100.0	100.0
middle	55	40	52	0.0	91.7	100.0	95.7
one	159	105	140	0.0	97.1	97.1	97.1
ready	184	100	166	0.0	100.0	98.5	99.25
rock	161	97	145	2.1	91.7	93.6	92.6
stop	105	56	99	0.0	100.0	100.0	100.0
two	139	98	131	0.0	97.0	97.0	97.0
	1158	685	1040	0.3	98.0	98.3	98.2
Ø (9 classes)				0.2	97.5	98.5	98.0

Figure 6.7: Classify results: Final hand gesture ANN after implementation

can not be improved more. These results lead with a mean recall of 97.5% and a precision of 98.5% which ends up with a F-score of 98%. Even these results are not as good as the results obtained in the final hand gesture ANN (see Figure 4.31) with a F-score of 98.6% which is 0.6% higher to the ones obtained in these last results (see Figure 6.7), the last ANN is more versatile since the environmental light, position of the camera and proximity of the user changes in the different images.

Task 1 ANN

To test the Task 1 ANN 74 images have been taken. Three colours have been used to do the markings of the industrial piece: black, the one used to train the ANN, red and blue. The obtained results with the different colour markings can be observed in Figure 6.9, Figure 6.10 and Figure 6.11. In Figure 6.8 the result of testing the industrial part without markings, the result of a good piece, can be observed.

If these images are observed, the ability of the ANN to search the markings and classify correctly the pieces, when the used pen is the blue or black, can be observed. Hence, when the red pen is used to make the markings, the ANN is not capable of finding the markings in the industrial part. Even though, the piece is not classified as a good piece either, which means it is not classified at all. As the ANN has only been trained with black markings, the difficulties of the ANN to search the markings in other

Figure 6.8: Good piece result of the test of the Task 1 ANN

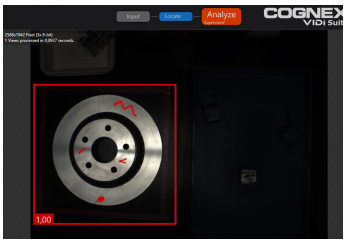


Figure 6.9: Result of the black marked Task 1 test

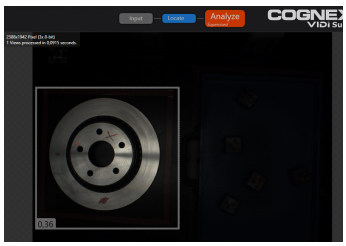


Figure 6.10: Result of the red marked Task 1 test

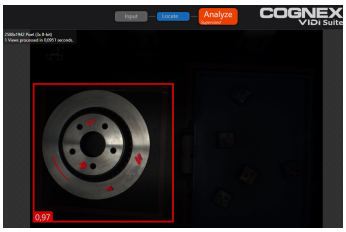


Figure 6.11: Result of the blue marked Task 1 test

colours is understandable. The ability of the ANN to search blue marks in the piece could be related to the similarity between black and dark blue colours.

Task 2 ANN

To test the Task 2 ANN 60 images have been taken. As in each image there are 5 dices, in total there are 300 tests of muffin and chihuahua images. These 60 images have been tested and all of them have been classified correctly. The obtained results with the different types of dices, chihuahuas and muffins, can be observed in Figure 6.12 and Figure 6.13. The chihuahua dices are classified as dogs and the muffins as muffins.

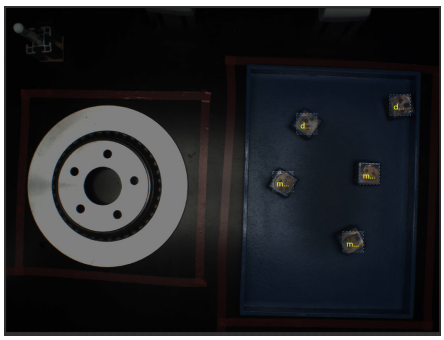


Figure 6.12: Task 2 ANN 1st test

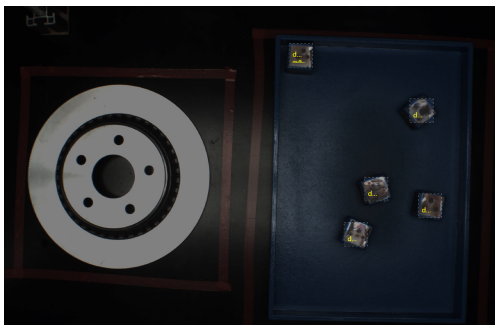


Figure 6.13: Task 2 ANN 2nd test

In these images, Figure 6.12 and Figure 6.13, the correct classification of the two types of dices is observable. On the one hand, when over the dice appears a *d*, means that the dice is classified as *dog*, hence, *chihuahua* dice. On the other hand, when the *m* letter appears over the dice, means that the dice is classifies as *muffin*.

6.2 The Reasoning and Execution Frameworks

In this project, the reasoning and execution frameworks are both placed in the robot. The reasoning framework is the one in charge of selecting the action to be applied depending the received input information. Whereas, the execution framework executes the action and specifies the robot which type of movement has to be done to which position. In this section, the implementation of these frameworks in the robot will be explained and tested. As the vision framework is still not connected to these two frameworks, the Hercules software will send the input information to the robot by the use of TCP/IP connection as it can be observed in Figure 6.14.

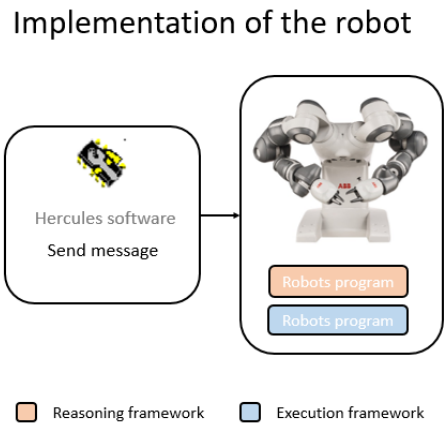


Figure 6.14: Implementation of the reasoning and execution frameworks

To make the TCP/IP connection between the robot and the computer, an Ethernet cable has been used. Apart from that, IP addresses have been established to enable this connection between the different devices as it can be seen in the Figure 6.15.

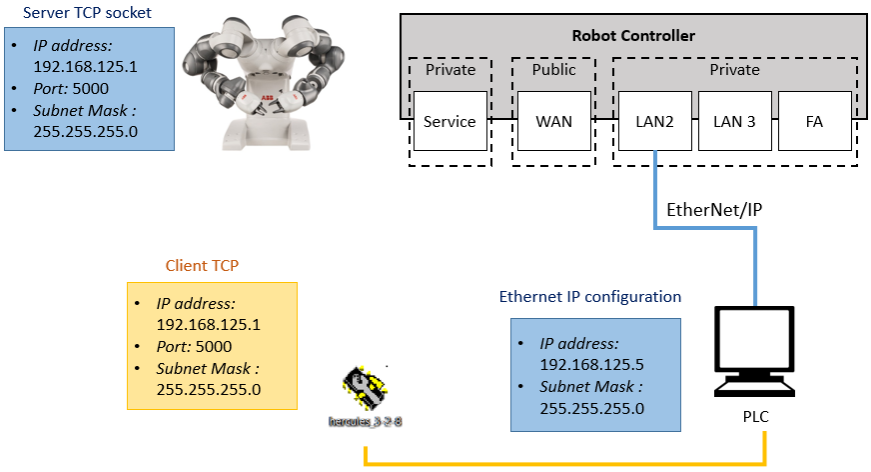


Figure 6.15: Network structure in the robot implementation phase

This implementation of the robot has been performed with the final robot arm programs, which means are different to the ones used for the simulation of the robot in Section 5 (see Figure 5.2 and Figure 5.3). Moreover, the part of the program which corresponds to the performance of the two tasks (see Section 3.3) has been changed. Even though, the emotion robot movements have not been changed from the previous

version. Therefore, by the implementation of these programs, emotion robot gestures (see Section 3.3.3) have been tested as well as the TCP/IP server connection and the two tasks movements with the real robot.

6.2.1 Flow Chart of the Robot Programs

In this subsection, the final flow charts of the robot arm programs are expressed as well as the last server `TCP_IP_ program`. In each section, the main module of each program is expressed. The data module of each process is the initialization of the global variables, this is expressed in the code in the annexes section.

TCP_IP_ program

This program creates the server socket to enable the communication between the robot and the vision program. As in this implementation of the robot program the vision part is not implemented, the gesture messages are sent to the robot by the Hercules software. To understand how this program works see the flow chart represented in Figure 6.16.

A remarkable point of this program is that it is not the final one since the robot works as server. When EAVS software has to communicate with the robot program the EAVS program works as server, so, the robot has to work as client. Due to this, for the hole implementation of the project, a last version of TCP_IP_ program, which will work as client, will be created.

The code of this program is placed in the appendices, the data module in the Appendix B.1.1 and the main module in the Appendix B.1.2.

T_ROB_R Right Robot Arm Program

The right robot arm orders over the left robot arm since the right arm has a more complex job than the left arm. This is due to the Task 1, where the right arm is the one in charge of all the movements. To understand how this program works see the flow chart represented in Figure 6.17.

The square in blue called *SyncTest* is a function, the explanation of it can be found in Section 5.1 and its flow chart in Figure 6.18. Moreover, the *CancelRountineR* is a routine connected to an interruption, which is given when cancel message is received (see Section 5.1). Apart from that, the green squares, except *HappyMoveR* and *SadMoveR*, are robot paths (see Section 5.1), the majority of them have simple move functions but *Pen_Path* and *BoardR_Path* have more complex codes since the gripper is controlled. In the case of the *HappyMoveR* and the *SadMoveR* are functions which establish the synchronization of both robot arms and after that the *HappyR* and *SadR* paths are run. To understand better each specific function, see Figure 6.18.

The code of this program is placed in the appendices, the data module in the Appendix B.2 and the

main module in the Appendix B.2.2.

T_ROB_L Left Robot Arm Program

In this section, the flow chart of the left robot arm program is expressed. This program and the T_ROB_R program are really similar since both arms of the robot have to do the same work pretty much. As mentioned in Section 6.2.1, the T_ROB_L is more simple than T_ROB_R since T_ROB_R is the head program which directs the actions of T_ROB_L. This can be observed in the flow chart in Figure 6.19). The functions of this program (see Figure 6.20) are really similar to the ones in Figure 6.18 too. The difference is that this program does not have a *Pen_Path* function since the left arm does not interact in Task 1.

The code of this program is placed in the appendices, the data module in the Appendix B.3.1 and the main module in the Appendix B.3.2.

6.2.2 Performance of the Robot

To see the performance of the robot with the programs explained in the flow charts Figure 6.16, Figure 6.17 and Figure 6.19 a video has been recorded. In this video, the process of sending the message of the hand gestures by TCP/IP server, the robot processing it and making the action can be observed (see `https://youtu.be/f9YFXDSfdT8`).

6.3 Complete Implementation

The workflow of the whole implementation can be observed in Figure 6.1. For this implementation, the TCP_IP_ program has been changed from server to client. The flow chart of this program can be observed in Figure 6.22 and the code is written in Appendix B.1.3. In this program, the client socket is made. Apart from that, the message code to be sent from the robot to EAVS program and vice versa has an specific structure which can be observed in Table 6.3. The specific messages to be sent between the software EAVS and the robot can be seen in Figure 6.21. This structure is to ensure the correct communication between EAVS and the robot program.

Sender message	Receiver answer
"ID + \0A"	"ID , nDataField1 + \0A"

Table 6.3: Structure of the message sent between the robot and EAVS program

There are two types of messages, one sent by the robot and the other one sent by EAVS software. On the one hand, the message sent by the robot has two part. First, the ID message which corresponds to the actual task (HAND, TASK1 or TASK2) and second, the "\0A" to point the end of the message. On the other hand, the message sent by EAVS is composed by three parts. First, the ID of the message,

in the second part of the message the result of the ANN is written and in the third part "\0A" is sent. Furthermore, for a more detailed information of the different messages see Figure 6.21.

As it can be observed in Figure 6.22, the TCP_IP_ client program, two new functions are used, the *mVision* EAVS initialization function (see Appendix B.1.3) and the communication function *mEAVS* (see Appendix B.1.3). The *mVision* function initializes the needed sockets to create the TCP/IP connections between the devices. Apart from that, the IP value and the port number are specified. The *mEAVS* function is more complex than the *mVision* function since is in charge of sending, processing and receiving the messages. To have a clearer idea of how this function works, observe the flow chart in Figure 6.23.

In *mEAVS* function, the robot message with the structure represented in Table 6.3 is sent and the answer of EAVS is received. Once the answer is obtained, the *mEAVS* function separates the three parts of the message and sends an array of two strings as result to the TCP_IP_ program. When the result is collected, the second string, hence the result of the ANN, is announced to the two robot arm programs. This way, the robot arms will activate to perform the commanded movement.

As mentioned before, the robot works as client and EAVS works as server[1]. There, the messages reported in Figure 6.21 are sent by the use of Hercules software, which works instead of EAVS software. The final step to complete the implementation of this project would be to run EAVS program in the computer and see the different parts of this project working together. To understand better the workflow of EAVS program see the hole implementation flow chart represented in Figure C.1, in Appendix C. The final implementation could not be proven since the person of the company in charge of the implementation of the ANN in EAVS software could not finish it on time.

[1]To check the TCP_IP_ client programs performance, visit `https://youtu.be/wrQlOaDWA7U`

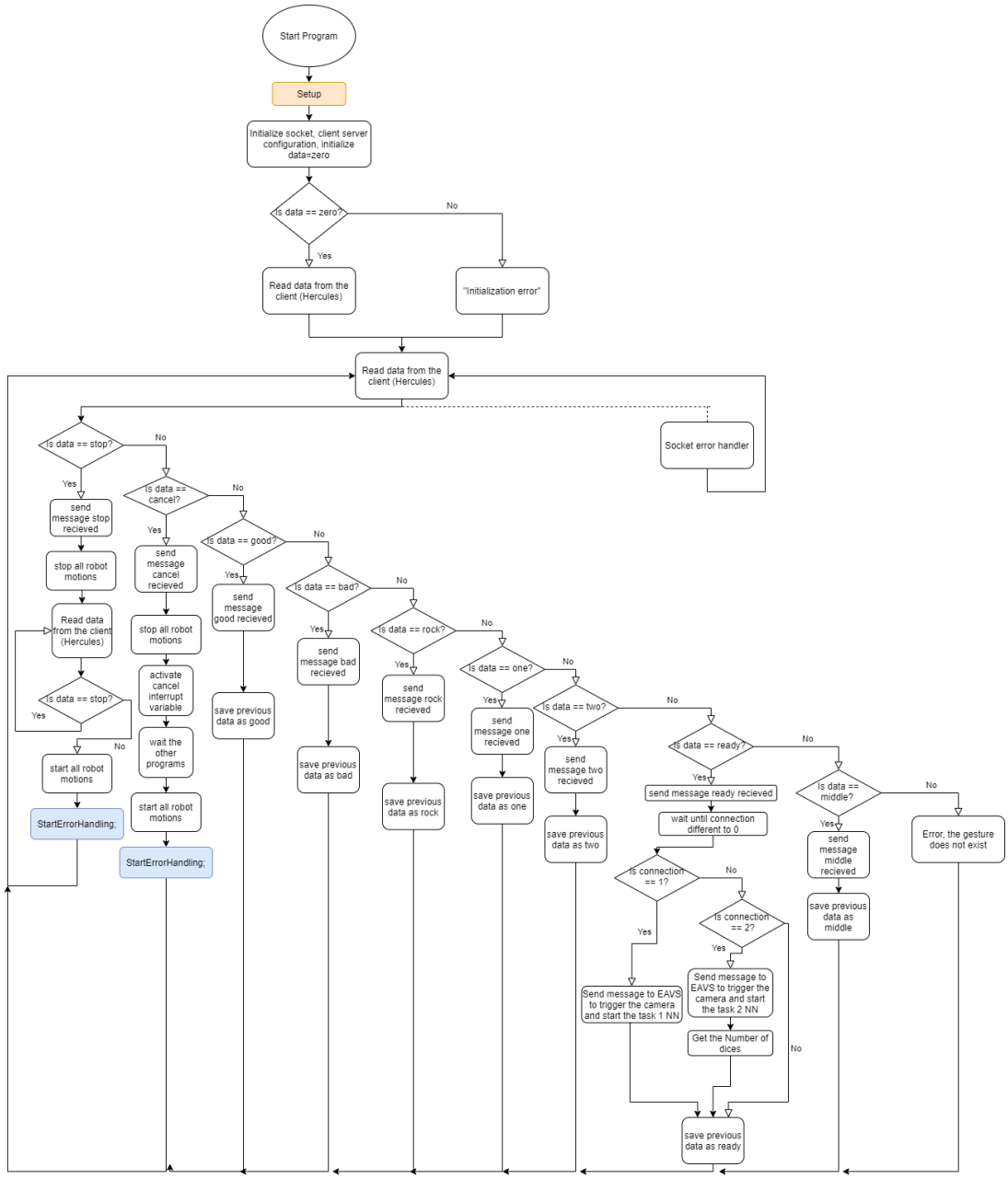


Figure 6.16: Flow chart of the TCP_IP_ server module

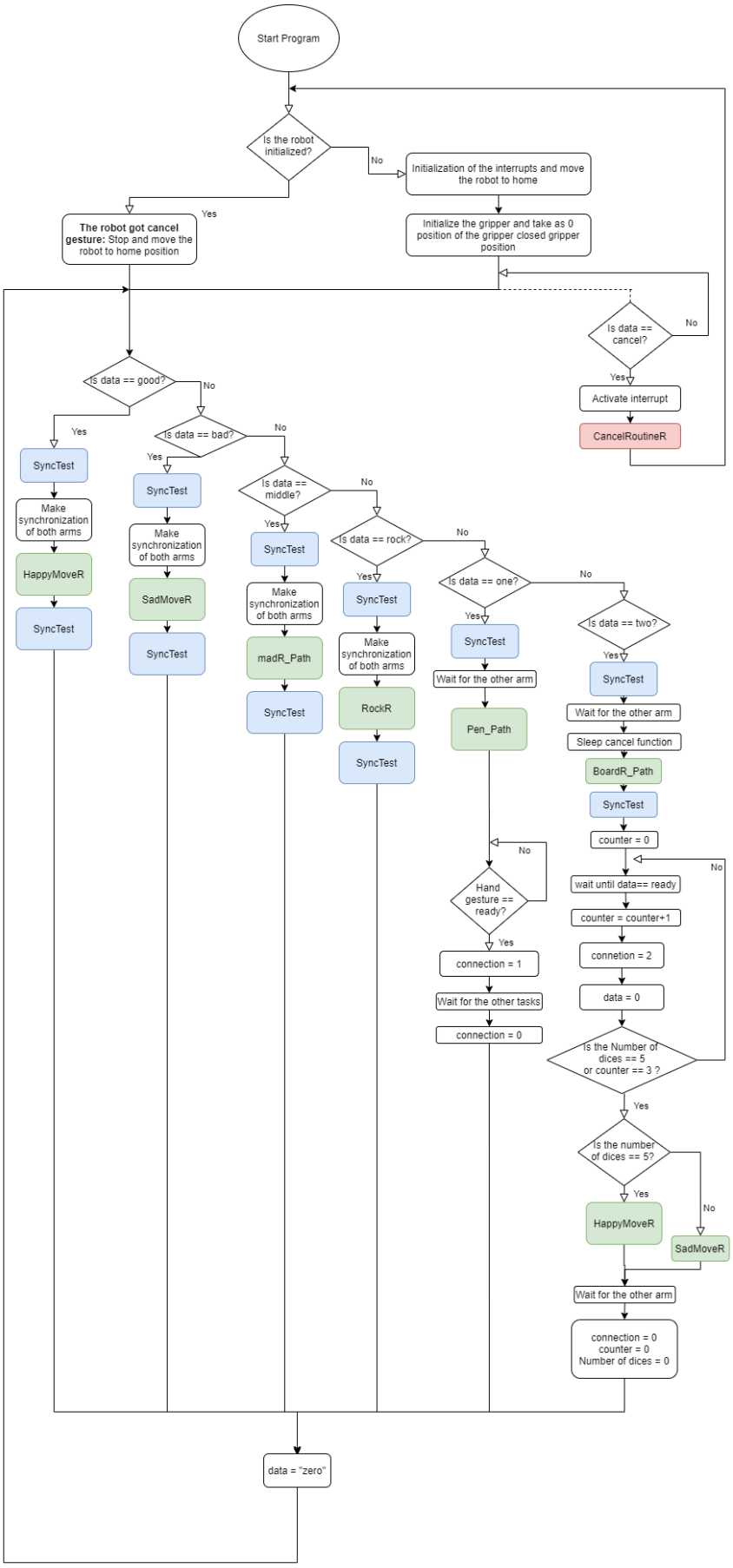


Figure 6.17: Final flow chart of the T_ROB_R module

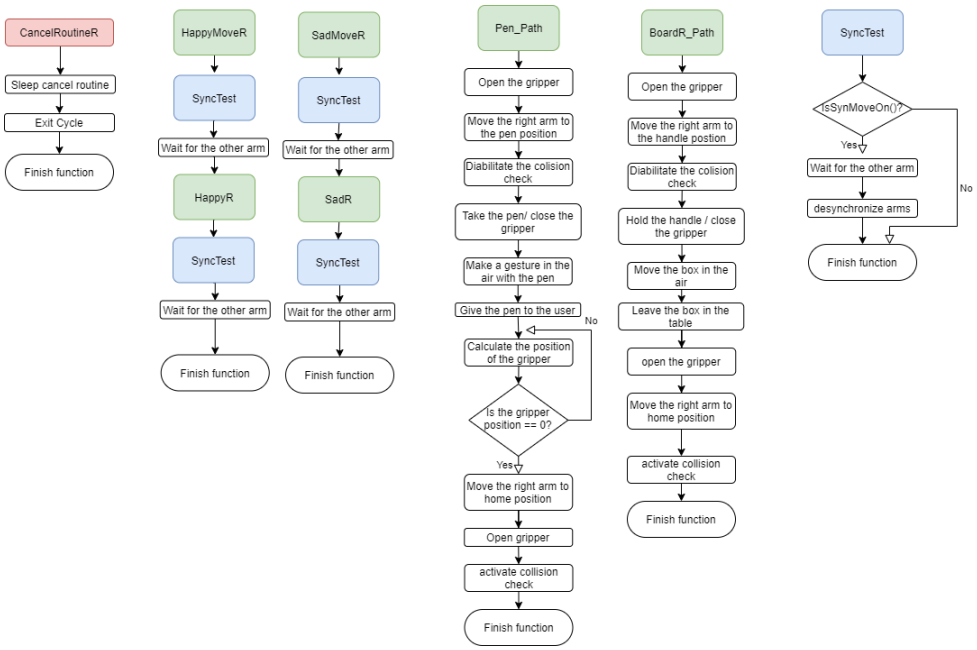


Figure 6.18: T_ROB_R functions

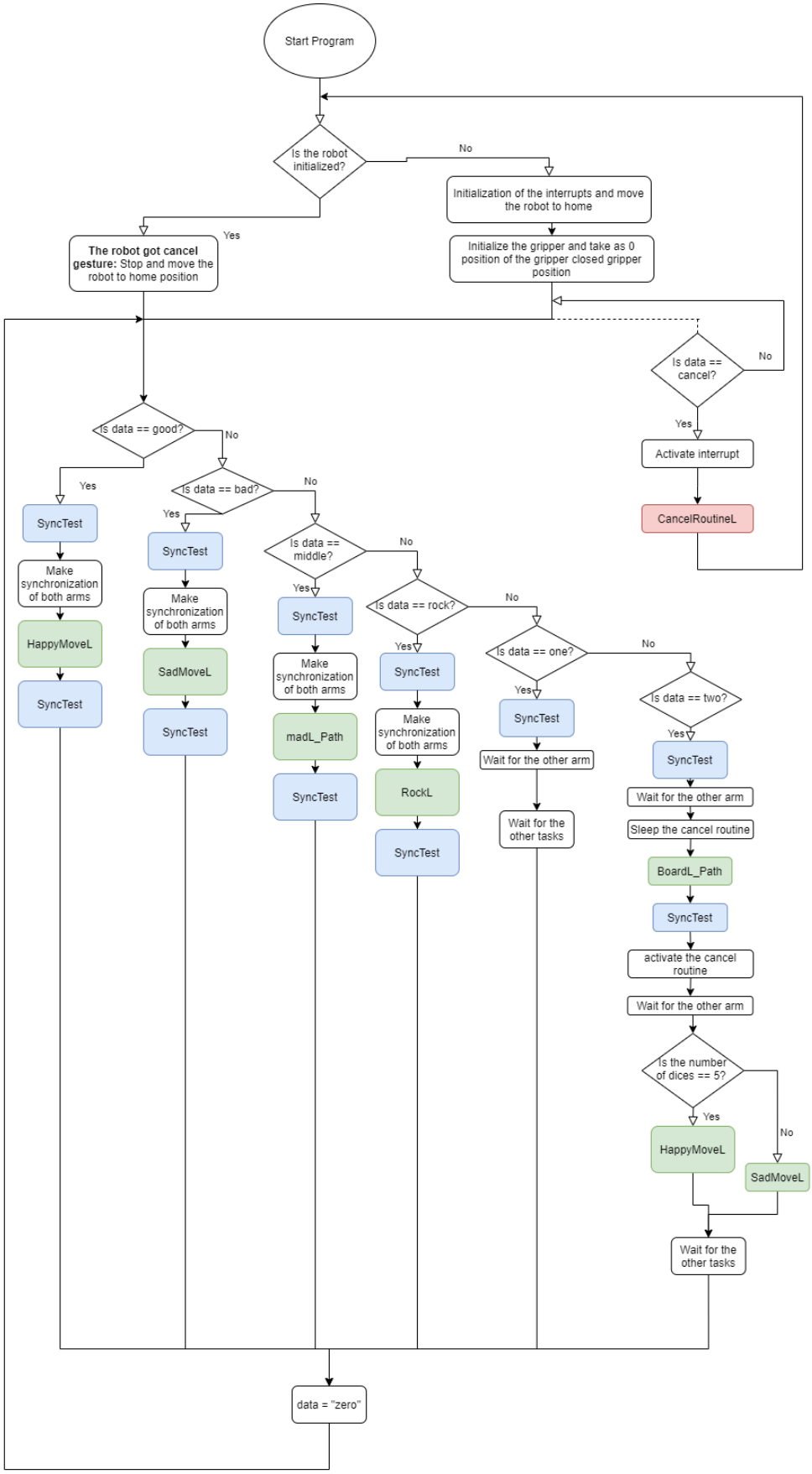


Figure 6.19: Final flow chart of the T_ROB_L module

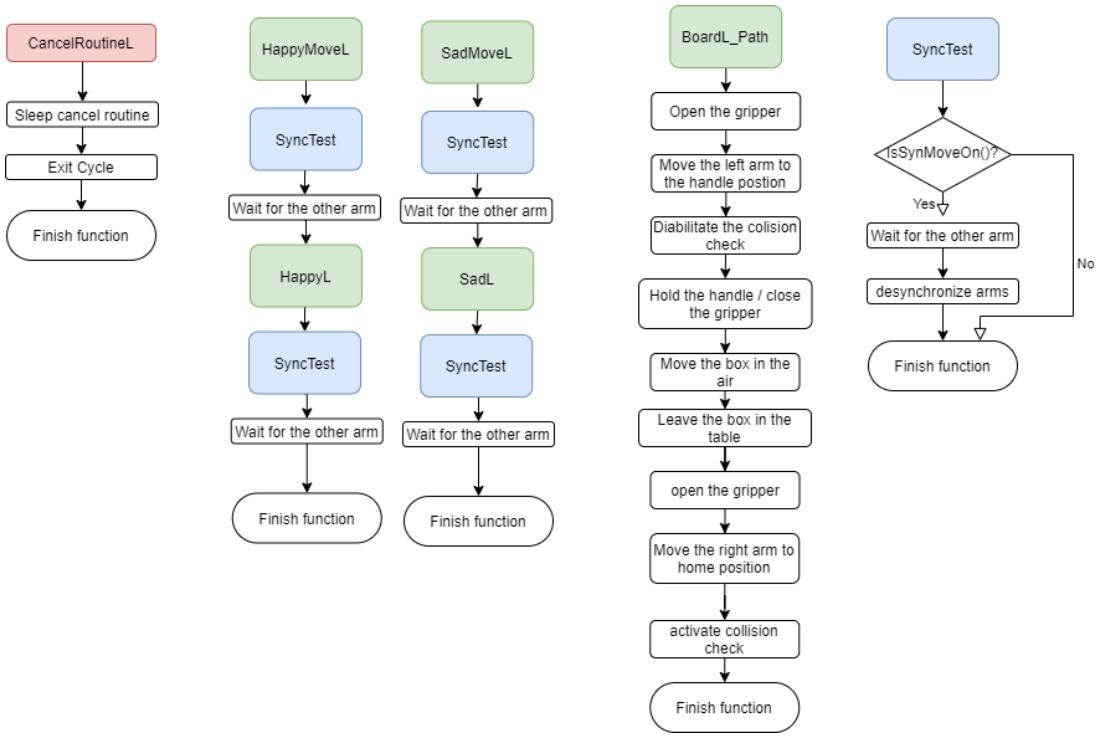


Figure 6.20: T_ROB_L functions

TCP/IP messages between the EAVS software and the robot

	Message	Sender message	Receiver answer
		Robot	EAVS
Hand gesture	good	"HAND\0A"	"HAND,good\0A"
	bad	"HAND\0A"	"HAND,bad\0A"
	stop	"HAND\0A"	"HAND,stop\0A"
	cancel	"HAND\0A"	"HAND,cancel\0A"
	ready	"HAND\0A"	"HAND,ready\0A"
	one	"HAND\0A"	"HAND,one\0A"
	two	"HAND\0A"	"HAND,two\0A"
	middle	"HAND\0A"	"HAND,middle\0A"
	rock	"HAND\0A"	"HAND,rock\0A"
	none	"HAND\0A"	"HAND,none\0A"
Task 1		"TASK1\0A"	"TASK1,ok\0A"
Task 2	0	"TASK2\0A"	"TASK2,0\0A"
	1	"TASK2\0A"	"TASK2,1\0A"
	2	"TASK2\0A"	"TASK2,2\0A"
	3	"TASK2\0A"	"TASK2,3\0A"
	4	"TASK2\0A"	"TASK2,4\0A"
	5	"TASK2\0A"	"TASK2,5\0A"

Figure 6.21: Messages between the robot and EAVS program

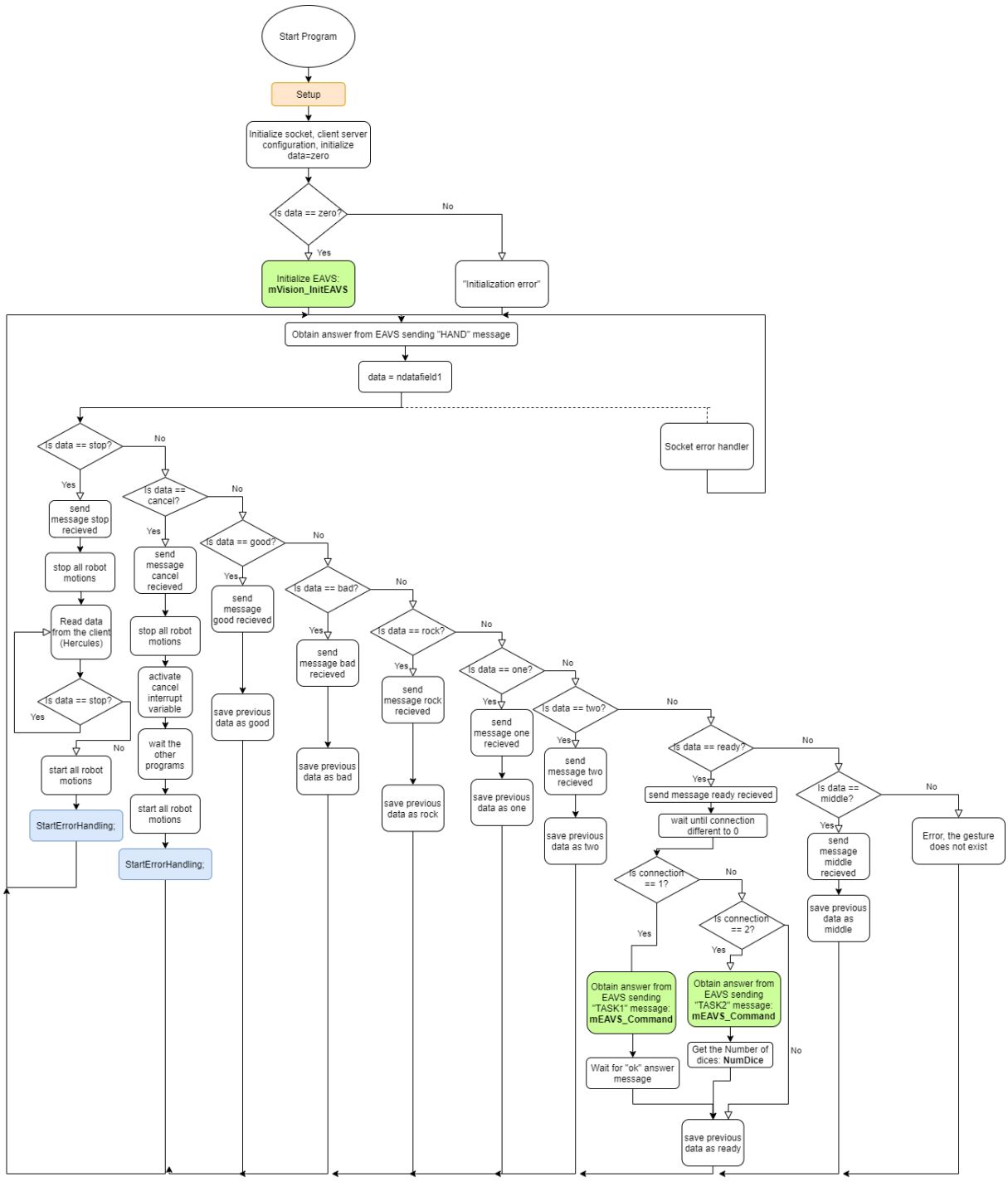


Figure 6.22: Flow chart of the TCP_IP_ client module

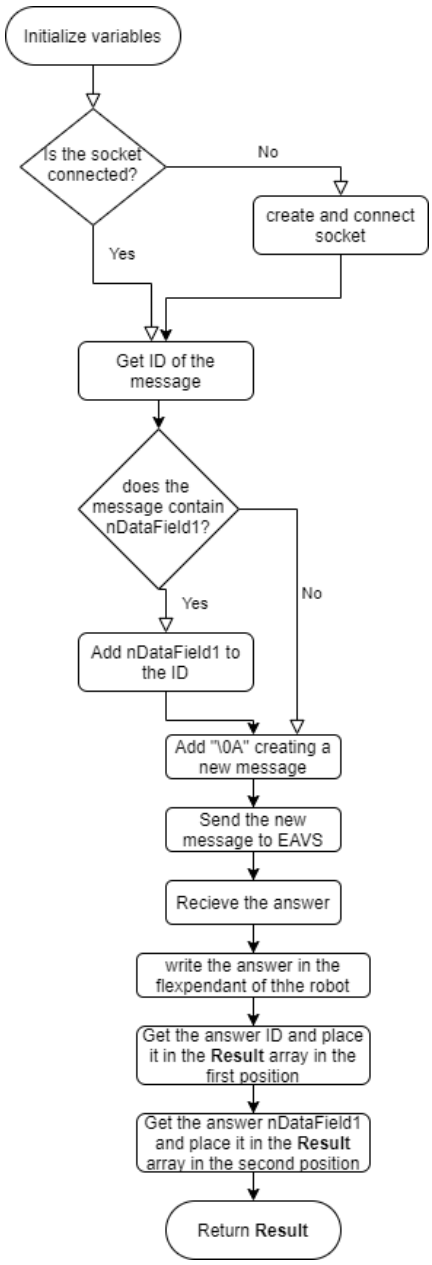


Figure 6.23: Flow chart of the mEAVS function

Chapter 7

Cost Planning

This project involves building some tools based on license paying programs, and testing them on a real robot. Hence, this project includes the development cost of the programs, the cost of the license and the different used machinery or equipment. The development cost of the programs has been computed based on an average of 8 hours work per day, and on an average, a junior developer quotes around 20 €/hour. The development cost of this project has been portrayed in Table 7.1.

To calculate the cost of the robot, the amortization of three years has been calculated. For that, the equation 7.1 has been used, where Vn is the value of the new robot, Vr the residual value (the value in which is expected to sell the robot), T the expected lifetime of the robot and H robots working hours per year. In this case, a residual value of 0 is taken since it is not expected to sell the robot after its usage. Furthermore, the lifetime of the robot is expected to be of 10 years and a usage of 800 hours/year.

$$A = \frac{\frac{(Vc-Vr)}{T}}{H} = \frac{\frac{(36{,}700-0)}{10}}{800} = 4.5875\, e/hour \tag{7.1}$$

The robot has already been used during 3 years with a usage of 800 hours/year, this makes a total amount of 2,400 hours in which the robot has been running. If the amortization of these 3 years, hence 2400 hours is taken, an actual cost of 11,010 € is obtained.

The development of the programs have been performed in a system with Intel Core i7 CPU @ 2.8GHz machine with 8GB of RAM with a graphical card of GeForce GTX 1050, has been utilised, the hardware might cost around 1.200,00 €.

Number of development days	124 days
Number of working hours	8.0 hours/day
Development wage	20 €/hour
Total development cost	19,840.00 €
Cost of YuMi ABB robot (estimated)	11,010 €
Cost of RobotStudio license	192.61 €
Cost of ViDi Cognex Software	13,964.57 €
Cost of computer for ViDi Cognex Software	3,370.76 €
Cost of Basler pia2400-17gc Camera	1,444.61 €
Cost of Basler acA2500-14gc Camera	866.77 €
Cost of Box for the dices, task 2	144.46 €
Cost of the dices	192.61 €
Cost of grippers and box handlers	385.23 €
Cost of industrial piece	4.82 €
Cost of the two lights	288.92 €
Cost of the structure of the project	529.69 €
Cost of the screen of the vision part	192.61 €
Total Cost	52,427.66 €

Table 7.1: Development cost for this project

Chapter 8

Sustainability Study

In this chapter, I will analyse the sustainability impact of this project from a view point as extended as possible. Sustainability is the satisfaction of current necessities without involving the capacity of future generations to satisfy theirs, guaranteeing the balance between the economical increase, environmental care and the social well-been. Sustainability and the sustainable development are two terms that are raising concern and are becoming to be linked to wider fields lately due to the population growth, which has led to a faster degradation of the environment and shortage of natural resources, in extreme cases [47].

In order to implement a sustainable artefact, the three fields shown in Figure 8.1, which are economic, social and environmental sustainability have been taken into account, as will be explained in the next sections. In this study, which consists in hand gesture recognition and its corresponding action performance by a YuMi robot, the development of an artefact will be done, as a proof of concept. For the development of this project there is no material to be chosen, so the sustainability of this project would be analysed as a service. This way, the service should respect the previously mentioned three sustainability types (see Figure 8.1), environmental, social and economic ones, in order to provide a sustainable solution.

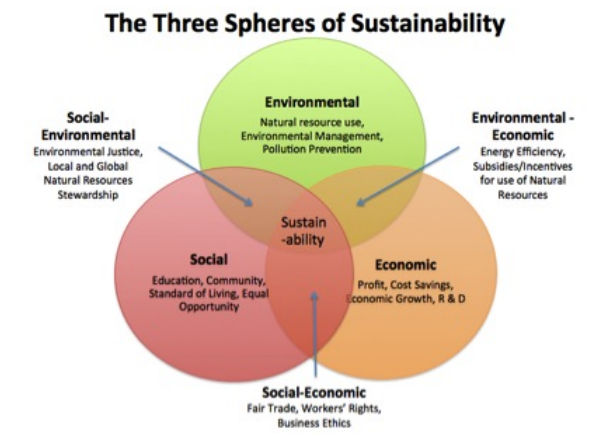


Figure 8.1: The three spheres of sustainability

8.1 Environmental

The environmental sustainability refers to the capacity of maintaining biological aspects in their productivity and variety over time. This way to ensure the preservation of the natural resources and to create awareness of the ecological aspects.

This project does not create any hazardous liquid, gas or material for the environment, hence it is not going to cause any kind of impact in the environment. For the creation and implementation of this project, computers, cameras and robots have been used. This machines are supplied by electricity, so the generation of the electricity needed to supply and the material needed to create these machines would be the only indirect impact in the environment caused by this project.

8.2 Social

The social sustainability refers to adopt values which generate behaviours as the value of nature, to maintain harmonic and satisfactory levels of education, training and awareness-raising for the population of a country to surpass themselves, to maintain a good living standard and to involve citizens to create something new for the society.

The social impact of this project is hard to estimate because depends in the mindset of the people, future industry situation and changes in technology. This project enables to introduce deep learning in industrial environment and shows how it could be applied. Apart from that, shows the option of controlling robots by hand gesture, which enables people without technical knowledge to be able to control the robot. This could have a really good acceptance in industrial environment but until its showing this can not be proven.

8.3 Economical

The economical sustainability refers to the capacity of generating wealth in suitable and equitable quantities in different social areas. It is main objective is that the population is capable of solving their economical issues as well as to strength the production and consumption in monetary sectors. To sum up, the economical sustainability concerns the balance between the human been and the nature to satisfy the necessities without putting at risk future generations.

When it comes to the economical impact of this project it is not easy to make any conclusion. As in the social sustainabilities the economical aspect of this project totally depends in the reaction of people, which can not be predicted. If the project and its idea is accepted and recognised could bring big profits but this totally depends in the economical situation, technology advance and in the mindset of the people and possible buyers.

Chapter 9

Conclusions

In conclusion, a framework capable of recognizing hand gestures and acting in answer to them has successfully been created. To do so, an ANN which is able to recognize hand gestures has been trained and tested. Furthermore, a YuMi robot has been programmed to accomplish the tasks selected by the hand gesture ANN. Apart from the objectives presented in the Section 1.2 two more ANN's have been trained, the industrial piece with markings and the chihuahua and muffin dices ANN.

Firstly, a study of the common knowledge hand gestures and human reactions has been performed to apply in this project. This way, an easy control of the robot is ensured since the input commands of the robot, the hand gestures, will be well known by the user.

For the training of the ANN's, ViDi software from Cognex has been used. The ANN of this software is a black box in which the only possible characteristics to be selected are the type of ANN (location, classification, read or analyze), feature size, color model and other image parameters, but the ANN structure, weights or activation function can not be modified. This brings a huge uncertainty in the performance of the ANN for different applications as well as in the manner of tuning the previously mentioned modifiable parameters. Due to this reason, an analysis of the software by the training of the hand gesture ANN has been completed. With this analysis, different aspects of the software have been clarified, as the non relation between the number of trained images and the given weight to a class. Another obtained conclusion is the need of more images in the training set when the identification window is increased in the location tool. This is due to the amount of data to be processed by the ANN, when the identification window is bigger, the ANN has more information to process so with more images better results are obtained. In addition, a bigger amount of images in the training of a class means better results in the recall and precision of that class. Another observed characteristic of this software, in the case of the classification tool, is that it reaches a point where more images in the training set confuses the ANN and the values of recall and precision decrease. When it comes to the hand gesture recognition, there is a tendency of miss-classification between the *one* and *two* gestures as well as between the *two* and *rock* gestures. Lastly, when the obtained results are compared with the number of images used with training purposes in each ANN, the ViDi software works better with industrial applications where the images have patterns in comparison to applications where the position or rotation of the object in track is more unpredictable.

When it comes to the programming, the YuMi robot has two arms and these are coded in separate programs. This means that when the objective is synchronized movement, the synchronization of both programs has to be ensured. To do so, both programs have to be in the same point of the code and after, the synchronization of both arms has to be created by a function in RAPID code. Apart from that, to make a TCP/IP connection in RobotStudio, the TCP/IP program has to be parallel to the robot arm programs. Otherwise, if the TCP/IP code is written in one of the robot arm programs as periodical interruption, so that the messages are obtained regularly from the other device, the robot looses a lot of information in the waiting time of the next interruption. Apart from that, returning to the previous point of the code, when the robot was in a synchronized movement, after the interruption is difficult. This is because when the interruption is given, the synchronization is lost and as the robot arm programs are parallel the communication between them is completely lost, hence the robot arms work individually. Furthermore, if the TCP/IP program is made in parallel, all these inconveniences are avoided since the TCP/IP program will be constantly executed and sending the information to the other programs by global variables.

Even though this project could not be shown and tested in the technological fair because of its cancellation due to COVID-19, the final implementation tests showed an easy controlability of the robot by simple commands. Apart from that, the results obtained in the performance of the three ANN are good since the hand gesture ANN F-score reached a performance of 98%, task 1 ANN with an AUC of 1 and task 2 ANN with a F-score of 98% and an AUC of 0.999. With this results it can be concluded that this project has been successful.

Chapter 10

Future improvements

Even though it has been proven that the project was successful, there are still some improvements, which are explained in the paragraphs bellow, that could be applied in future work.

Firstly, the implementation of the different parts of the project, hence the robot framework and the performance of the trained ANN, could be set together by the implementation of the EAVS software. This could not be implemented in this project because of external factors. Nevertheless, the flow chart of the entire implementation can be observed in Figure C.1.

Secondly, for a better interaction between the robot and the user, sounds and facial expressions can be added to the robot. As the YuMi robot is not equipped with speakers or any sound system, an external device would have to be used. Furthermore, a study of this case has been done in this project and a possible solution has been found. Hence, an Arduino board able to read SD cards could be used next to a sound amplifier and an external speaker. By the use of YuMi robots digital outputs, signals to the Arduino board could be sent to play sounds saved on the SD card. This solution can be observed in D.1. Another way to improve the interaction between the user and the robot is by adding facial expression to the robot. To do so, a screen would have to be added to the robot and programming of user interfaces (UI's). An example of possible facial expressions for the robot can be seen in D.2.

Thirdly, for a more advanced communication between the user and the robot, more hand gestures and robot reactions could be added to the project. Nevertheless, this change could have a bad effect since more hand gestures and reactions lead to a more complex interaction which could end up confusing the user.

Lastly, the performance of two of the ANN's, the hand gesture and the chihuahua and muffins ANN's, can be improved. Better results could be reached by the addition of images to the training set until obtaining results as close as possible to a perfect performance, that is to say 100% of F-score or an AUC of 1.

Acknowledgements

The research and work behind this project was carried out during the spring semester of the year 2020 in Elektroautomatik (EA) company of Skövde, Sweden. First and foremost, I gratefully acknowledge the support of my university supervisor Cecilio Angulo for his encouraging, great support and patience throughout the duration of this project.

In addition, I want to thank my two company supervisors Samuel and Adam for their support, guidance, advice and for bringing the opportunity to perform this project in your company.

Furthermore, I wish to express my sincere thanks to Elektroautomatik, for providing me with all the necessary facilities for the research. This work was financed by EA, through the project for the technological fair of Elmia.

Moreover, I would like to express my gratitude to Fredrik Wahlström, Daniel Bohrén and Calle Mellergård computer vision professionals, for their help regarding vision recognition technologies and help with the ViDi software. I would also want to thank Martin Skoog for his support within the robotics programming area.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

Last but not least, I would like to thank my family and friends for supporting me in general, and in my work and studies in particular. Thank you all for being there at all times.

References

[1] Oates Briony J. *Researching information systems and computing.* Sage Publications, March 23 2006. URL https://www.rnibbookshare.org/browse/book/1207589?returnPath%5Cu003dL2Jyb3dzZS9hdXRob3I/bW9kdWxlTmFtZT1wdWJsaWMma2V5PUJyaW9ueSBKLiBPYXRlcw==.

[2] Angkoon Phinyomark Kyrre Glette Erik Scheme François Laviolette Ulysse Côté-Allard, Gabriel Gagnon-Turcotte and Benoit Gosselin. Unsupervised Domain Adversarial Self-Calibration for Electromyographic-based Gesture Recognition. *arXiv:1912.11037v1*, 2019. URL https://arxiv.org/pdf/1912.11037.pdf.

[3] Kale Geetanjali Vinayak and Patil Varsha Hemant. A Study of Vision based Human Motion Recognition and Analysis. *International Journal of Ambient Computing and Intelligence*, 7, 2016. URL https://arxiv.org/ftp/arxiv/papers/1608/1608.06761.pdf.

[4] Pabbaraju Chirag Shaheer Nadeem Simhadri Harsha Vardhan Seshadri Vivek Varma Manik Patil Shishir G., Dennis don Kurian and Jain Pratee. GesturePod: Enabling On-device Gesture-based Interaction for White Cane Users. *Microsoft Research India*, 2019. URL http://manikvarma.org/pubs/patil19.pdf.

[5] Zhang Daqing Qi Guande Wu Jiahui, Pan Gang and Li Shijian. Gesture Recognition with a 3-D Accelerometer. *Springer, Berlin, Heidelberg*, 2009. URL https://link.springer.com/chapter/10.1007/978-3-642-02830-4_4.

[6] Pascal Schneider, Raphael Memmesheimer, Ivanna Kramer, and Dietrich Paulus. Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping. *arXiv:1906.12171v1*, 2019. URL https://2019.robocup.org/downloads/program/SchneiderEtAl2019.pdf.

[7] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. *arXiv:1704.02463v2*, pages 409–419, 06 2018. doi: 10.1109/CVPR.2018.00050. URL https://www.researchgate.net/publication/329745590_First-Person_Hand_Action_Benchmark_with_RGB-D_Videos_and_3D_Hand_Pose_Annotations/citation/download.

[8] Prashan Premaratne. Chapter 2. In *Historical Development of Hand Gesture Recognition*, 2017.

[9] G.R.S Murthy and R.S. Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, pages 405–410, July-December 2009. URL https://www.researchgate.net/publication/228659202_A_review_of_vision_based_hand_gesture_recognition.

[10] Siddharth S. Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: A survey. *Artif. Intell. Rev.*, 43(1):1–54, January 2015. ISSN 0269-2821. doi: 10.1007/s10462-012-9356-9. URL `https://doi.org/10.1007/s10462-012-9356-9`.

[11] Ram Pratap Sharma and Gyanendra K. Verma. Human computer interaction using hand gesture. *Procedia Computer Science 54*, pages 721–727, 2015. URL `https://core.ac.uk/download/pdf/82501716.pdf`.

[12] Okan Köpüklü, Neslihan Köse, and Gerhard Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2184–21848, 2018.

[13] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. Vision based hand gesture recognition. *World Academy of Science, Engineering and Technology*, pages 972–977, 2009.

[14] G. Devineau, F. Moutarde, W. Xi, and J. Yang. Deep learning for hand gesture recognition on skeletal data. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 106–113, May 2018. doi: 10.1109/FG.2018.00025.

[15] Leonid Sigal, Stan Sclaroff, and Vassilis Athitsos. Skin color-based video segmentation under time-varying illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:862–877, 2004.

[16] Monowar Hossain and Michael Jenkin. Recognizing hand-raising gestures using hmm. In *Proceedings of the 2nd Canadian Conference on Computer and Robot Vision*, CRV '05, page 405–412, USA, 2005. IEEE Computer Society. ISBN 0769523196. doi: 10.1109/CRV.2005.67. URL `https://doi.org/10.1109/CRV.2005.67`.

[17] Pavel Senin. Dynamic time warping algorithm review. Technical report, Information and Computer Science Department, University of Hawaii, 2008.

[18] Holzmann GJ. *Finite state machine*. Harvard University, 1925. URL `http://www.spinroot.com/spin/Doc/Book91_PDF/F1.pdf`.

[19] Okan Kopuklu, Ahmet Gunduz, Neslihan Köse, and Gerhard Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. In *IEEE International Conference on Automatic Face and Gesture Recognition*, page 8, 02 2019. URL `https://www.researchgate.net/publication/331134940_Real-time_Hand_Gesture_Detection_and_Classification_Using_Convolutional_Neural_Networks`.

[20] Rama Chellappa Yi-Tong Zhou. *Artificial Neural Networks for Computer Vision*, volume 5. Springer-Verlag New York, 1992. ISBN 978-1-4612-2834-9. doi: 10.1007/978-1-4612-2834-9.

[21] Richard J. Mammone. *Artificial neural networks for speech and vision*. 1994.

[22] C. Pattichis, Christodoulos Christodoulou, Efthyvoulos Kyriacou, and Marios Pattichis. Artificial neural networks in medical imaging systems. In *Proceedings 1st MEDINF International Conference on Medical Informatics and Engineering*, pages 83–91, 10 2003.

[23] V. C. Patel, R. W. McClendon, and J. W. Goodrum. Color computer vision and artificial neural networks for the detection of defects in poultry eggs. *Artificial Intelligence Review*, 12:163–176, 1998.

[24] R. Parisi, E. D. Di Claudio, G. Lucarelli, and G. Orlandi. Car plate recognition by neural networks and image processing. In *1998 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pages 195–198, 1998. ISBN 0-7803-4455-3. doi: 10.1109/ISCAS.1998.703970.

[25] Vijay Laxmi and Harish Rohil. License plate recognition system using back propagation neural network. *International Journal of Computer Applications*, 99:29–37, 08 2014. doi: 10.5120/17395-7945.

[26] Dan Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *International Joint Conference on Artificial Intelligence IJCAI-2011*, pages 1237–1242, 07 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-210.

[27] Kıvanç Kılıç, Ismail Boyacı, Hamit Köksel, and İsmail Küsmenoğlu. A classification system for beans using computer vision system and artificial neural networks. *Journal of Food Engineering*, 78: 897–904, 02 2007. doi: 10.1016/j.jfoodeng.2005.11.030.

[28] C.-C Yang. Application of artificial neural networks in image recognition and classification of crop and weeds. *Canadian Biosystems Engineering / Le Genie des biosystems au Canada*, 42:147–152, 01 2000.

[29] WiredWorkers. From robot to cobot, a look through history, 2016. URL `https://wiredworkers.io/from-robot-to-cobot/`.

[30] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human-robot collaboration: a survey. *I. J. Humanoid Robotics*, 5:47–66, 03 2008. doi: 10.1142/S0219843608001303.

[31] Jonas Tjomsland, Ali Shafti, and Aldo Faisal. Human-robot collaboration via deep reinforcement learning of real-world interactions, 12 2019.

[32] Philip Long, Chevallereau Christine, Damien Chablat, and Alexis Girin. An industrial security system for human-robot coexistence. *Industrial Robot: An International Journal*, page 8, 12 2017. doi: 10.1108/IR-09-2017-0165.

[33] George Michalos, S. Makris, Panagiota Tsarouchi, Toni Guasch, Dimitris Kontovrakis, and George Chryssolouris. Design considerations for safe human-robot collaborative workplaces. *Procedia CIRP*, 37:248–253, 12 2015. doi: 10.1016/j.procir.2015.08.014.

[34] Johan Kildal, Alberto Tellaeche, Izaskun Fernández, and Iñaki Maurtua. Potential users' key concerns and expectations for the adoption of cobots. *Procedia CIRP*, 72:21 – 26, 2018. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2018.03.104. URL `http://www.sciencedirect.com/science/article/pii/S2212827118302592`. 51st CIRP Conference on Manufacturing Systems.

[35] Åsa Fast-Berglund, Filip Palmkvist, Per Nyqvist, Sven Ekered, and Magnus Åkerman. Evaluating cobots for final assembly. *Procedia CIRP*, 44:175 – 180, 2016. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2016.02.114. URL `http://www.sciencedirect.com/science/article/pii/S2212827116003966`. 6th CIRP Conference on Assembly Technologies and Systems (CATS).

[36] N. Keerthi Shravani and Sri. BVS Rao. Introducing robots without creating fear of unemployment and high cost in industries. *International Journal of Engineering Technology Science and Research*, 5:11, 01 2018. ISSN 2394 – 3386. URL `http://www.ijetsr.com/images/short_pdf/1518956866_1128-1135-ieteh312_author_will_send_updated_paper_etsr.pdf`.

[37] Ana M. Djuric, R.J. Urbanic, and J.L. Rickli. A framework for collaborative robot (cobot) integration in advanced manufacturing systems. *SAE International Journal of Materials and Manufacturing*, 9 (2):457–464, 2016. ISSN 19463979, 19463987. URL `http://www.jstor.org/stable/26267460`.

[38] D. Surdilovic, G. Schreck, and U. Schmidt. Development of collaborative robots (cobots) for flexible human-integrated assembly automation. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, June 2010.

[39] Maj Stenmark. *Intuitive Instruction of Industrial Robots : A Knowledge-Based Approach*. PhD thesis, Lund University, 05 2017. URL `https://lup.lub.lu.se/search/ws/files/24801702/espik.pdf`.

[40] Maj Stenmark, Elin Anna Topp, Mathias Haage, and Jacek Malec. Knowledge for synchronized dual-arm robot programming. In *Papers from the 2017 AAAI Fall Symposia*. AAAI Press, 2017. ISBN 978-1-57735-794-0.

[41] David Kirschner, Rosemarie Velik, Saeed Yahyanejad, Mathias Brandstötter, and Michael Hofbaur. Yumi, come and play with me! a collaborative robot for piecing together a tangram puzzle. In Andrey Ronzhin, Gerhard Rigoll, and Roman Meshcheryakov, editors, *Interactive Collaborative Robotics*, pages 243–251, Cham, 2016. Springer International Publishing. ISBN 978-3-319-43955-6.

[42] Maj Stenmark, Andreas Stolt, Elin A. Topp, Mathias Haage, Anders Robertsson, Klas Nilsson, and Rolf Johansson. The giftwrapper: Programming a dual-arm robot with lead-through. In *Human-Robot Interfaces for Enhanced Physical Interactions : ICRA 2016 Full-day Workshop*, 2016.

[43] Raji Ridwan Adetunji and Koh Phel Sze. Understanding Non-Verbal Communication across Cultures: A Symbolic Interactionism Approach. *Kluwer Academic Publishers-Plenum Publishers*, 7, 2012. URL `https://link.springer.com/article/10.1023/A:1024716331692`.

[44] Dane Archer. Unspoken Diversity: Cultural Differences in Gestures. *i-Come International Conference on Communication and Media 2012, Penang, Malaysia, 1-3 November*, 18, 1997. URL `http://qualquant.org/wp-content/uploads/video/1997%20Archer79-105.pdf`.

[45] ITing Huang. Finger-counting around the world, 2016. URL `https://termcoord.eu/2016/09/finger-counting-around-the-world/`.

[46] Azlinah H Mohamed Michael W. Berry and Bee Wah Yap. *Supervised and Unsupervised Learning for Data Science*. Springer International Publishing, 2020. URL `https://books.google.se/books?id=0mGtDwAAQBAJ&printsec=frontcover&dq=unsupervised+and+supervised+learning+article&hl=en&sa=X&ved=0ahUKEwif3rHRm4vpAhXLy6YKHb09B9AQ6AEIJzAA#v=onepage&q=unsupervised%20and%20supervised%20learning%20article&f=false`.

[47] Bob Willard. 3 Sustainability Models, 2010. URL `https://sustainabilityadvantage.com/2010/07/20/3-sustainability-models/`.

[48] T. Fukuda, J. Taguri, F. Arai, M. Nakashima, D. Tachibana, and Y. Hasegawa. Facial expression of robot face for human-robot mutual communication. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 46–51, 2002. ISBN 0-7803-7272-7. doi: 10.1109/ROBOT.2002.1013337.

[49] M. Han, C. Lin, and K. Song. Autonomous emotional expression generation of a robotic face. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 2427–2432, 2009.

Appendix A

The training phase

A.1 Task 1 ANN

In this section of the appendix, the results obtained in the tests made with different feature sizes (see Table 4.11) are shown in case a more detailed information is wanted to be known.

A.2 Task 2 ANN

In this section of the appendix, the results obtained in the tests made with different feature sizes (see Table 4.13) are shown in case a more detailed information is wanted to be known.

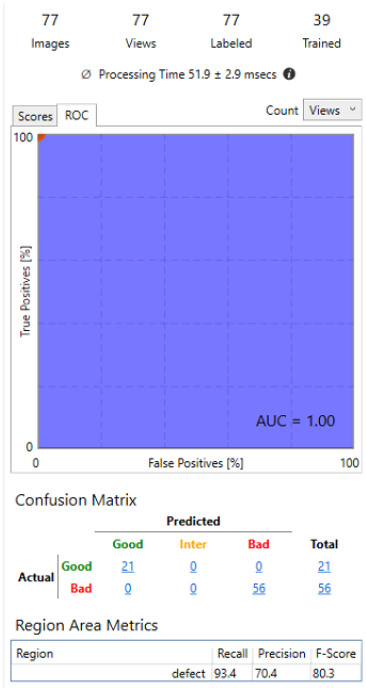


Figure A.1: 3rd approach: ROC curve

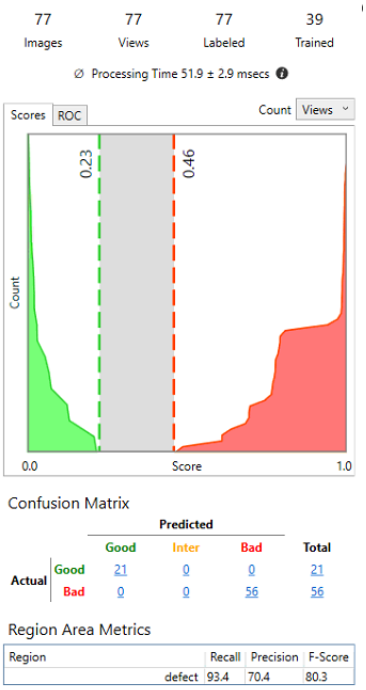


Figure A.2: 3rd approach: Scores graphic

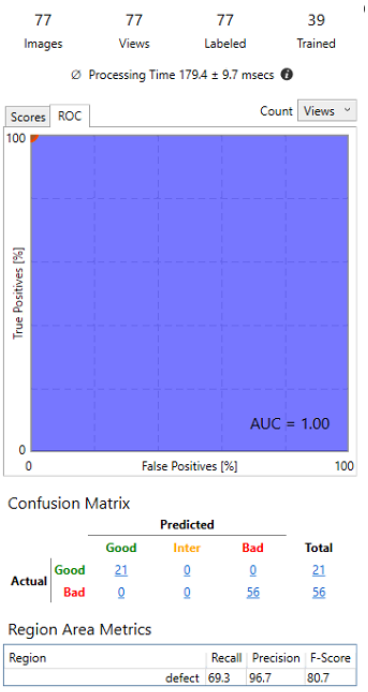


Figure A.3: 4rth approach: ROC curve

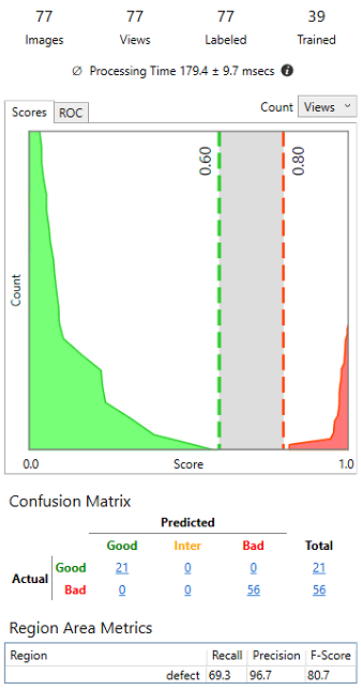


Figure A.4: 4rth approach: Scores graphic

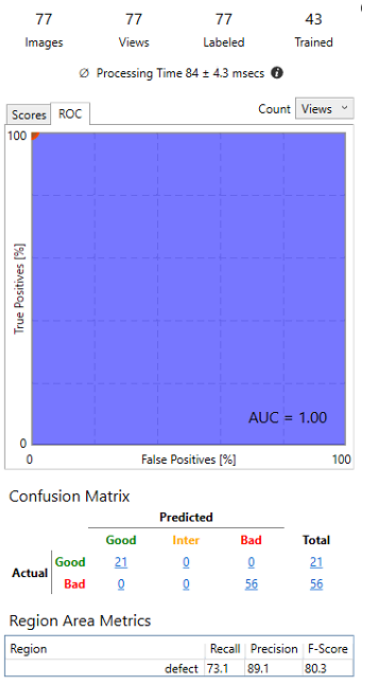


Figure A.5: 5th approach: ROC curve

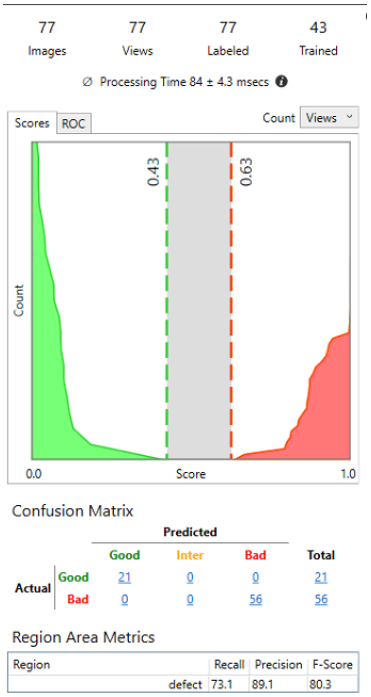


Figure A.6: 5th approach: Scores graphic

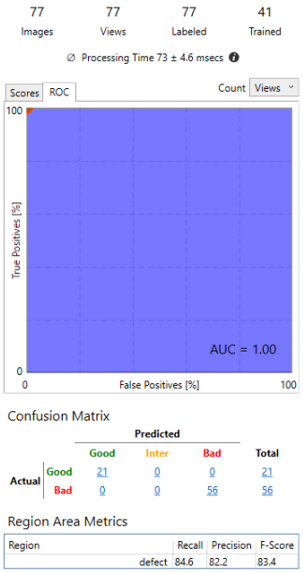


Figure A.7: 6th approach: ROC curve

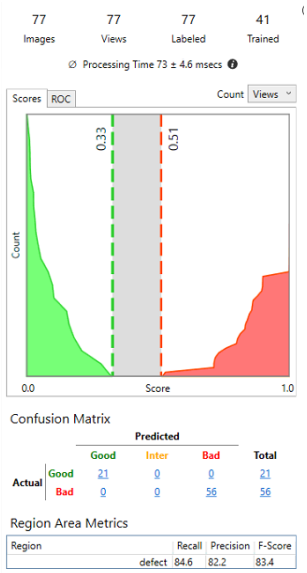


Figure A.8: 6th approach: Scores graphic

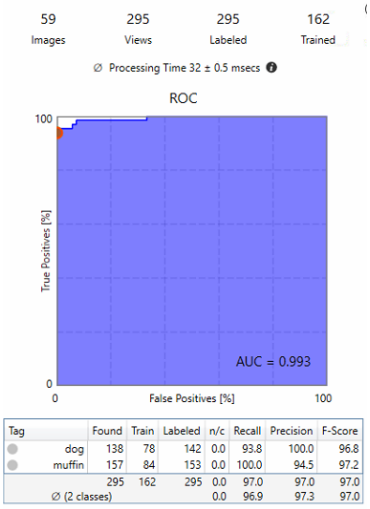


Figure A.9: 2nd approach: ROC curve

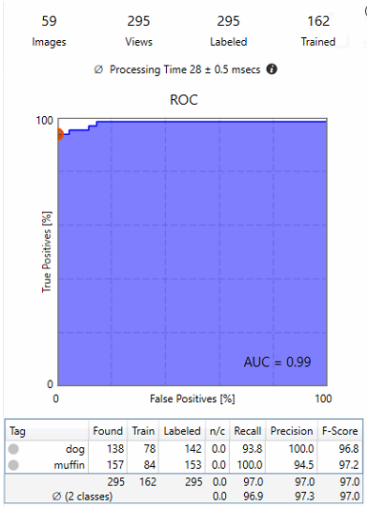


Figure A.10: 3rd approach: ROC curve

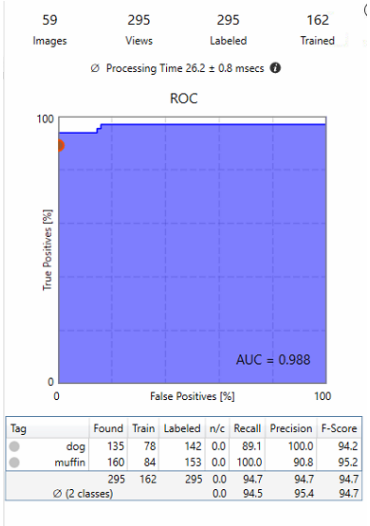


Figure A.11: 4^{rth} approach: ROC curve

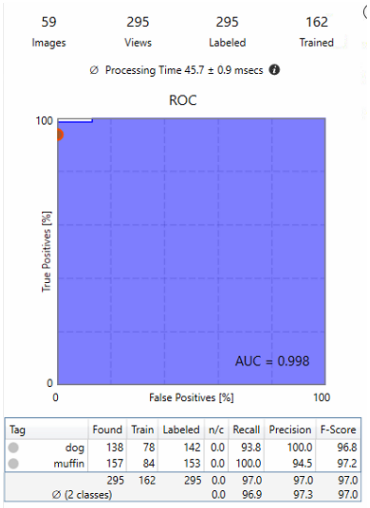


Figure A.12: 5th approach: ROC curve

Appendix B

Robot Program Code

B.1 TCP IP Code

B.1.1 Data Module

```
MODULE DataModule

    !
    ***********************************************************************
    !Shared data RobL-RobR-TCP_IP_Input
    !
    ***********************************************************************
    !interrupts
    PERS bool InitDone := FALSE;!START IN FALSE
    PERS bool MoveCanceled := FALSE;
    PERS string data;
    PERS intnum connection:=0;
    PERS intnum NumDice;
ENDMODULE
```

B.1.2 Server

Main Module

```
MODULE MainModule
    PERS tasks TaskListThree{3}:=[["T_ROB_L"],["T_ROB_R"],["TCP_IP_Input"]];
```

```
3
4       VAR syncident syncTaskDefaultStart;
5       VAR syncident syncTaskFinishStop;
6       VAR syncident syncTaskFinishCancel1;
7       VAR syncident syncTaskFinishCancel2;
8       VAR syncident syncTaskFinishCancel3;
9       VAR syncident syncTaskFinishOne;
10      !TCP/ip
11      VAR socketdev serverSocket;
12      VAR socketdev clientSocket;
13      VAR socketstatus statusTCP;
14      VAR string dataRecieve;
15      VAR string NumDiceRecieve;
16      VAR string dataPrev;
17
18
19      PROC main()
20          data:="zero";
21          !TCP/ip initialization/connection
22          SocketCreate serverSocket;
23          SocketBind serverSocket,"192.168.125.1",5000;
24          SocketListen serverSocket;
25          SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
26          SocketSend clientSocket\Str:="connected";
27
28          !take first data
29          IF data="zero" THEN
30              SocketReceive clientSocket\Str:=dataRecieve;
31              data:=dataRecieve;
32              dataPrev:=data;
33          ELSE
34              SocketSend clientSocket\Str:="initialization error";
35          ENDIF
36          WaitSyncTask syncTaskDefaultStart,TaskListThree;
37          InitDone:=TRUE;
38          WHILE TRUE DO
39
40              SocketReceive clientSocket\Str:=dataRecieve;
41              data:=dataRecieve;
42
43              IF (data="stop") AND (NOT dataPrev="two") THEN
44                  SocketSend clientSocket\Str:=" stop received";
45                  ! tell the client stop has been recieved
46                  StopMove\Quick\AllMotionTasks;
47                  !while is the stop gesture mantain in loop
48                  WHILE data="stop" DO
49                      !StopMove\Quick;
50                      SocketReceive clientSocket\Str:=dataRecieve;
51                      SocketSend clientSocket\Str:=" data received";
52                      data:=dataRecieve;
```

```
                ENDWHILE
                StartMove\AllMotionTasks;
                StartErrorHandling;

            ELSEIF data="cancel" THEN
                dataPrev:="cancel";
                SocketSend clientSocket\Str:=" cancel received";
                StopMove\Quick\AllMotionTasks;
                !semaphore to activate Right arm interrupt
                IF MoveCanceled=TRUE THEN
                    MoveCanceled:=FALSE;
                ELSEIF MoveCanceled=FALSE THEN
                    MoveCanceled:=TRUE;
                ENDIF
                WaitSyncTask syncTaskFinishCancel1,TaskListThree;
                StartMove\AllMotionTasks;
                StartErrorHandling;
                WaitSyncTask syncTaskFinishCancel2,TaskListThree;
                WaitSyncTask syncTaskFinishCancel3,TaskListThree;

            ELSEIF data="good" THEN
                dataPrev:="good";
                SocketSend clientSocket\Str:=" good received";

            ELSEIF data="bad" THEN
                dataPrev:="bad";
                SocketSend clientSocket\Str:=" bad received";

            ELSEIF data="rock" THEN
                dataPrev:="rock";
                SocketSend clientSocket\Str:=" rock received";

            ELSEIF data="one" THEN
                dataPrev:="one";
                SocketSend clientSocket\Str:=" one received";

            ELSEIF data="two" THEN
                dataPrev:="two";
                SocketSend clientSocket\Str:=" two received";

            ELSEIF data="ready" THEN
                SocketSend clientSocket\Str:=" ready received";
                !connection gives the information of which task has been
    previously selected, if connecction is = 1 is task 1 otherwise is task 2
                WaitUntil connection<>0;
                ! that means a task has been selected
                IF connection=1 THEN
                    SocketSend clientSocket\Str:="task 1";
                    !send a message to EAVS of selection of task 1
                    WaitSyncTask syncTaskFinishOne,TaskListThree;
```

```
                ELSEIF connection=2 THEN
                    SocketSend clientSocket\Str:="task 2";
                    !In case of the second task, look how many dices have been counted by EAVS
                    SocketReceive clientSocket\Str:=NumDiceRecieve;
                    IF NumDiceRecieve="1" THEN
                        NumDice:=1;
                    ELSEIF NumDiceRecieve="2" THEN
                        NumDice:=2;
                    ELSEIF NumDiceRecieve="3" THEN
                        NumDice:=3;
                    ELSEIF NumDiceRecieve="4" THEN
                        NumDice:=4;
                    ELSEIF NumDiceRecieve="5" THEN
                        NumDice:=5;
                    ENDIF
                ENDIF
                data:="zero";


            ELSEIF data="middle" THEN
                SocketSend clientSocket\Str:=" middle received";

                !in case the user makes something that it is not considered a gesture
            ELSEIF data<>"stop" AND data<>"cancel" AND data<>"zero" THEN
                SocketSend clientSocket\Str:=" Error, incorrect entrance:";
                SocketSend clientSocket\Str:=dataRecieve;
                !clean the variables
                data:="zero";
                dataRecieve:="";
            ENDIF
        ENDWHILE

    !error handling for TCP/IP
    ERROR
        !to notify if there is any error the next
        IF ERRNO=ERR_SOCK_TIMEOUT THEN
            RETRY;
        ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
            SocketClose clientSocket;
            SocketClose serverSocket;
            SocketCreate serverSocket;
            SocketBind serverSocket,"192.168.125.1",5000;
            SocketListen serverSocket;
            SocketAccept serverSocket,clientSocket,\Time:=WAIT_MAX;
            RETRY;
        ELSE
            RAISE ;
        ENDIF
```

```
    ENDPROC


    PROC StartErrorHandling()
    ERROR
        IF ERRNO=ERR_STARTMOVE THEN
            WaitTime 1;
            StartMove\AllMotionTasks;
        ELSEIF ERRNO=ERR_PROGSTOP THEN
            !execution stopped several times
        ELSEIF ERRNO=ERR_ALRDY_MOVING THEN
            !robot already moving
        ENDIF
    ENDPROC

    PROC setup()
        !Initialize variables
        data:="zero";
        InitDone:=FALSE;

    ENDPROC
ENDMODULE
```

B.1.3 Client

Main Module

```
MODULE MainModule
    PERS tasks TaskListThree{3}:=[["T_ROB_L"],["T_ROB_R"],["TCP_IP_Input"]];
    PERS tasks TaskListTask2{2}:=[["TCP_IP_Input"],["T_ROB_R"]];

    VAR syncident syncTaskDefaultStart;
    VAR syncident syncTaskFinishStop;
    VAR syncident syncTaskFinishCancel1;
    VAR syncident syncTaskFinishCancel2;
    VAR syncident syncTaskFinishCancel3;
    VAR syncident syncTaskFinishOne;
    VAR syncident syncTaskTwoDice;
    !TCP/ip
    VAR socketdev serverSocket;
    VAR socketdev clientSocket;
    VAR socketstatus statusTCP;
    VAR string dataRecieve;
    VAR string NumDiceRecieve;
    VAR string dataPrev;

```

```
!EA Vision Studio results
VAR recEAVSResult Result;

!Procedure alarm numbers
VAR errnum errCommandEAVS:=1;

PROC main()
    data:="zero";
    InitDone:=FALSE;
    !initialize EAVS
    mVision_InitEAVS;
    WaitSyncTask syncTaskDefaultStart,TaskListThree;
    InitDone:=TRUE;
    WHILE TRUE DO

     !Send command to EA Vision Studio

    Result:=mEAVS_Command(EAVS1,"HAND", errCommandEAVS);
    GetEAVSHandMessage(Result);

        IF (data="stop") AND (NOT dataPrev="two") THEN
            ! tell the client stop has been recieved
            StopMove\Quick\AllMotionTasks;
            !while is the stop gesture mantain in loop
            WHILE data="stop" DO
                !StopMove\Quick;
                 Result:=mEAVS_Command(EAVS1,"HAND", errCommandEAVS);
                 GetEAVSHandMessage(Result);
            ENDWHILE
            StartMove\AllMotionTasks;
            StartErrorHandling;

        ELSEIF data="cancel" THEN
            dataPrev:="cancel";
            StopMove\Quick\AllMotionTasks;
            !semaphore to activate Right arm interrupt
            IF MoveCanceled=TRUE THEN
                MoveCanceled:=FALSE;
            ELSEIF MoveCanceled=FALSE THEN
                MoveCanceled:=TRUE;
            ENDIF
            WaitSyncTask syncTaskFinishCancel1,TaskListThree;
            StartMove\AllMotionTasks;
            StartErrorHandling;
            WaitSyncTask syncTaskFinishCancel2,TaskListThree;
            WaitSyncTask syncTaskFinishCancel3,TaskListThree;

        ELSEIF data="good" THEN
            dataPrev:="good";

```

```
            ELSEIF data="bad" THEN
                dataPrev:="bad";

            ELSEIF data="rock" THEN
                dataPrev:="rock";

            ELSEIF data="one" THEN
                dataPrev:="one";

            ELSEIF data="two" THEN
                dataPrev:="two";

            ELSEIF data="ready" THEN
                !connection gives the information of which task has been
    previously selected, if connecction is = 1 is task 1 otherwise is task 2
                WaitUntil connection<>0;
                ! that means a task has been selected
                IF connection=1 THEN
                     mVision_InitEAVS;
                     Result:=mEAVS_Command(EAVS1,"TASK1", errCommandEAVS);
                     WaitUntil result.nResultID="TASK1" AND result.
    nDatafield1="ok";
                    !send a message to EAVS of selection of task 1
                    WaitSyncTask syncTaskFinishOne,TaskListThree;
                ELSEIF connection=2 THEN
                     mVision_InitEAVS;
                     Result:=mEAVS_Command(EAVS1,"TASK2", errCommandEAVS);
                     GetEAVSTaskMessage(Result);
                    !In case of the second task, look how many dices have
    been counted by EAVS
                    WaitSyncTask syncTaskTwoDice,TaskListTask2;
                ENDIF
                data:="zero";

            ELSEIF data="middle" THEN
                dataPrev:="two";
                !in case the user makes something that it is not considered a
     gesture
            ELSEIF data<>"stop" AND data<>"cancel" AND data<>"zero" THEN
                SocketSend clientSocket\Str:=" Error, incorrect entrance:";
                SocketSend clientSocket\Str:=dataRecieve;
                !clean the variables
                data:="zero";
                dataRecieve:="";
            ENDIF
        ENDWHILE


    ENDPROC

```

```
116
117     PROC StartErrorHandling()
118     ERROR
119         IF ERRNO=ERR_STARTMOVE THEN
120             WaitTime 1;
121             StartMove\AllMotionTasks;
122         ELSEIF ERRNO=ERR_PROGSTOP THEN
123             !execution stopped several times
124         ELSEIF ERRNO=ERR_ALRDY_MOVING THEN
125             !robot already moving
126         ENDIF
127     ENDPROC
128
129     PROC GetEAVSHandMessage(recEAVSResult result)
130
131       IF result.nResultID="HAND" AND result.nDatafield1="good" THEN
132               data:="good";
133       ENDIF
134
135       IF result.nResultID="HAND" AND result.nDatafield1="bad" THEN
136             data:="bad";
137       ENDIF
138
139       IF result.nResultID="HAND" AND result.nDatafield1="middle" THEN
140               data:="middle";
141       ENDIF
142
143       IF result.nResultID="HAND" AND result.nDatafield1="rock" THEN
144             data:="rock";
145       ENDIF
146
147             IF result.nResultID="HAND" AND result.nDatafield1="one" THEN
148               data:="one";
149       ENDIF
150
151       IF result.nResultID="HAND" AND result.nDatafield1="two" THEN
152             data:="two";
153       ENDIF
154
155       IF result.nResultID="HAND" AND result.nDatafield1="stop" THEN
156               data:="stop";
157       ENDIF
158
159       IF result.nResultID="HAND" AND result.nDatafield1="cancel" THEN
160             data:="cancel";
161       ENDIF
162         IF result.nResultID="HAND" AND result.nDatafield1="ready" THEN
163             data:="ready";
164       ENDIF
165
```

```
    ENDPROC

        PROC GetEAVSTaskMessage(recEAVSResult result)

      IF result.nResultID="TASK2" AND result.nDatafield1="0" THEN
              NumDice:=0;
      ENDIF

      IF result.nResultID="TASK2" AND result.nDatafield1="1" THEN
            NumDice:=1;
      ENDIF

      IF result.nResultID="TASK2" AND result.nDatafield1="2" THEN
              NumDice:=2;
      ENDIF

      IF result.nResultID="TEASK2" AND result.nDatafield1="3" THEN
           NumDice:=3;
      ENDIF

            IF result.nResultID="TASK2" AND result.nDatafield1="4" THEN
              NumDice:=4;
      ENDIF

      IF result.nResultID="TASK2" AND result.nDatafield1="5" THEN
            NumDice:=5;
      ENDIF

    ENDPROC

    PROC setup()
        !Initialize variables
        data:="zero";
        InitDone:=FALSE;

    ENDPROC
ENDMODULE
```

mEAVS

```
MODULE mEAVS


! Record for camera settings
  RECORD recEAVS
    socketdev ComSocket;
    num nPortNumber;
    string stIPNumber;
```

```
ENDRECORD

!EAVSCommand results
RECORD recEAVSResult
  string nResultID;
  string nDataField1;
ENDRECORD

!***********************************************************************
!  PROCEDURE: mEAVS_Command
!  Code writer: Hans Nordberg
!  Date: 2011-10-18
!
!  Parameters: ComSocket,nCommandID,nDataField1-10,escapeErrCommand,
 nStatusComID,nStatusComID,escapeErrOnline
!
  !  PROCEDURE: mEAVS_Command
!  Modified by: Leire Amenabar
!  Date: 2020-06-05
!
!  Added Parameters: nLength
  !  Eliminated parameters: nDataField2-10
!
!
!***********************************************************************
FUNC recEAVSResult mEAVS_Command(
  VAR recEAVS EAVS,
  string nCommandID,
  \string nDataField1,
  errnum escapeError
  )

  !*****- Function variables -*****

  !Result
  VAR recEAVSResult Result;

  !Timeout SocketSend
  VAR bool bOnline:=FALSE;
  !Time values of the socket
  VAR num nSockRecTimeOut:=50;
  VAR num nSockConTimeOut:=5;
  !Variable of the hole message to be sent to EA Vision Studio
  VAR string stCommand;
  !Variable for the result
  VAR string stResult;
  !Variables for the recieving message
  VAR num nStartPos:=0;
  VAR bool bDone:=FALSE;
      VAR intnum nLength;
```

```
!*****- Control the socket status with EA Vision Studio -*****

!Socket status control
IF SocketGetStatus(EAVS.Comsocket) <> SOCKET_CONNECTED THEN
  !Connect the robot with EA Vision Studio
  SocketClose EAVS.ComSocket;
  SocketCreate EAVS.ComSocket;
  SocketConnect EAVS.ComSocket,EAVS.stIPNumber,EAVS.nPortNumber\Time:=
nSockConTimeOut;
ENDIF

!*****- Send message to EA Vision Studio -*****

!Take the ID of the message to send
stCommand:=nCommandID;

!In case has a field add the field
IF present (nDataField1) THEN
  stCommand:=stCommand+","+ nDataField1;
ENDIF


!Add the ending of the message
stCommand:=stCommand+"\0A";

!Send the hole message to the computer
SocketSend EAVS.ComSocket \Str:=stCommand;

!*****- Recieving message from EA Vision Studio -*****

!Recieve message from EA Vision Studio
    SocketReceive EAVS.ComSocket\Str:=stResult\Time:=nSockRecTimeOut;
    TPWrite "recieved" + stResult; ! Prints the result in the flex
pendant

bDone:=FALSE;
!bDone:=StrToVal(StrPart(stResult,1,4),Result.nResultID); ! converts from
 string to val (HAND)

!Exempel recieved= HAND,good\0A
!Exempel Result=["HAND","good"]

nStartPos:=StrMatch(stResult,1,",")+1; !position of the character that is
 after the first comma  StrMatch(stResult,1,",") -> StrMatch(message,
starting position,finding character)
nLength:=(StrMatch(stResult,nStartPos,"0")-2)-(nStartPos-1); ! the length
 of the word after the first ","
    !-> find the 0 character position -2 (\0A) to get the last character
position of the word after the comma => HAND,good\0A => d character
```

```
position in this example
        ! -nStartPos => the position of the first letter of the word after the comma -1-> to get the length of the word that is after the first comma.

        Result.nResultID:=StrPart(stResult,1,StrMatch(stResult,1,",")-1); ! Give the result ID by taking the string part (StrPart) of the stResult (HAND,good\0A) from the first charachter (H) with the length of from the beggining to the character before the comma (D)-> length of 4
    Result.nDataField1:=StrPart(stResult,nStartPos,nLength);!Take the string part of the stResult (HAND,good\0A) from thhe nStartPos with the length of nLength

    !Return the result
    RETURN Result;

    ERROR
      !in case of error error handling
      RAISE escapeError;
  ENDFUNC

ENDMODULE
```

mVision

```
MODULE mVision


    !****************************************
    !*****- Camera settings -*****
    !****************************************

    !EA Vision Studio
    VAR recEAVS EAVS1;
    PERS string nEAVSStation:="10";

    !*****************************************
    !*****- Vision function's response data -*****
    !*****************************************


    !*********************************************************************
    !  PROCEDURE: mVision_initEAVS
    !  Author: Hans Nordberg
    !  Date: 2011-10-19
    !
    !  PROCEDURE: mVision_initEAVS
```

```
  !  Modified by: Leire Amenabar
  !  Date: 2020-06-05
    !
    !  Description: Connects robot to EAVS
    !
    !
   ***********************************************************************
    PROC mVision_InitEAVS()
        !Procedure alarm numbers
        VAR errnum errCommandEAVS:=1;

        !Setup EA Vision Studio
        !
        EAVS1.nPortNumber:=2001;
        EAVS1.stIPNumber:="192.168.125.5";
        nEAVSStation:="10";

        !Status check EA Vision Studio
       ! mEAVS_StatusCheck EAVS1,nEAVSStation,"100","200",errCommandEAVS;

    ERROR
        TEST ERRNO
        CASE errCommandEAVS:
           ! ReportAndAbort "EAVS: statuscheck timeout eller fel kommando";
        ENDTEST
    ENDPROC
ENDMODULE
```

B.2 Right robot arm program Code

B.2.1 Data Module

```
MODULE DataModule

    !*******************************************************
    !Shared data RobL-RobR-TCP_IP_Input
    !*******************************************************
    !interrupts
    PERS bool InitDone;
    PERS bool MoveCanceled;
    PERS string data:="zero";
    PERS intnum connection:=0;
    PERS intnum NumDice;

    !*******************************************************
    ! Rob_R data
```

```
    !*********************************************************
    PERS tooldata Servo:=[TRUE
    ,[[0,0,114.2],[1,0,0,0]],[0.215,[8.7,12.3,49.2],[1,0,0,0]
    ,0.00021,0.00024,0.00009]];
  TASK PERS wobjdata RobtR:=[FALSE,TRUE,""
    ,[[0,0,0],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];

ENDMODULE
```

B.2.2 Main Module

```
MODULE MainModule
    !*********************************************************
    !
    ! Module:  Module1
    !
    ! Description:
    !   <Insert description here>
    !
    ! Author: leire amenabar
    !
    ! Version: 1.0
    !
    !*********************************************************
    ! Shared variables by T_ROB_L and T_ROB_R
    PERS tasks TaskListDefaultPos{2}:=[["T_ROB_L"],["T_ROB_R"]];
    PERS tasks TaskListTask2{2}:=[["TCP_IP_Input"],["T_ROB_R"]];
    PERS tasks TaskListThree{3}:=[["T_ROB_L"],["T_ROB_R"],["TCP_IP_Input"]];
    !wait sync ident
    VAR syncident syncTaskDefaultStart;

    VAR syncident syncTaskReadyGood;
    VAR syncident syncTaskReadyBad;
    VAR syncident syncTaskReadyOne;
    VAR syncident syncTaskReadytwo;
    VAR syncident syncTaskReadyMiddle;
    VAR syncident syncTaskReadyRock;

    VAR syncident syncTaskFinishGood;
    VAR syncident syncTaskFinishBad;
    VAR syncident syncTaskFinishOne;
    VAR syncident syncTaskFinishtwo;
    VAR syncident syncTaskFinishMiddle;
    VAR syncident syncTaskFinishRock;

    VAR syncident syncTaskFinishCancel1;
    VAR syncident syncTaskFinishCancel2;
```

```
    VAR syncident syncTaskFinishCancel3;
    VAR syncident syncTaskCancel;

    VAR syncident syncTaskRockStop;

    VAR syncident syncTaskTwo;
    VAR syncident syncTaskTwoDice;

    !synchro making ident
    VAR syncident syncMoveOnGood;
    VAR syncident syncMoveOnBad;
    VAR syncident syncMoveOnStop;
    VAR syncident syncMoveOnMiddle;
    VAR syncident syncMoveOnTaskTwo;

    VAR syncident syncMoveCancel;
    VAR syncident syncCancel;

    !Interrupts
    VAR intnum persStopIntL;
    VAR intnum persCancelintL;

    !Variables
    VAR BOOL CancelCycle:=FALSE;
    VAR intnum counter:=0;
    VAR num nRightHandPos;
    VAR num nRightHandState;

    !Interrupts
    VAR intnum persStopInt;
    VAR intnum persCancelint;


    !RobotTargets
    CONST robtarget Target_10_1:=[[50.93727774,-211.58009673,-57.503310529],
    [0.681705621,-0.317739889,0.422922566,0.505425871],[0,0,0,4],
    [-101.964426493,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_10:=[[394.552488228,213.326359531,-64.691837231],
    [0.707107107,-0.000001182,-0.000000583,-0.707106455],[0,0,0,0],
    [-146.664815972,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[394.55261883,70.452812463,-65.823829472],
    [0.707107107,-0.000001182,-0.000000583,-0.707106455],[0,0,0,0],
    [-146.664815972,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget HappyR_10:=[[60.957925302,445.607575047,456.449117885],
    [0.071967237,0.070402452,-0.088760676,-0.990951943],[1,0,0,4],
    [68.620858195,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget HappyR_20:=[[60.955515952,503.645940808,523.610550629],
    [0.061023532,0.052122332,0.023208561,-0.996504267],[1,0,0,4],
    [103.5993711,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget HappyR_30:=[[60.95528943,503.645611603,672.110740551],
[0.053747759,0.059596985,0.152261621,-0.985076533],[1,0,-1,4],
[125.21597527,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget SadR_10:=[[54.944541606,539.031416692,380.398676486],
[0.445042891,-0.48119534,-0.592982499,-0.467717463],[1,3,2,4],
[39.105047487,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget SadR_20:=[[54.945330891,632.632107318,400.458332345],
[0.445043108,-0.481195875,-0.592981981,-0.467717364],[1,3,2,4],
[52.457904341,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget madR_10:=[[-105.817477025,182.610106295,198.627859614],
[0.492434949,0.113516942,-0.860686719,0.061968514],[0,0,0,4],
[-101.964436053,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget madR_20:=[[-105.817626662,182.61026717,198.627919452],
[0.652124321,0.099108607,-0.746999117,0.083088347],[0,0,0,4],
[-101.96442831,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget madR_30:=[[-105.817738328,182.61053665,198.627927454],
[0.773068217,0.082993786,-0.620998653,0.099187881],[0,0,0,4],
[-101.964427601,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Home_R:=[[9.578368507,182.609892723,198.627808149],
[0.523068661,0.111214912,-0.842420918,0.066010726],[0,0,0,4],
[-101.964427132,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Pen_10
:=[[-172.93,336.58,260.03],[0.0578979,0.731557,-0.677851,-0.0446021],
[0,0,-1,5],[-155.948,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_30
:=[[-172.92,336.60,155.70],[0.0578581,0.731514,-0.677903,-0.0445885],
[0,0,-1,5],[-155.947,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_40
:=[[-172.93,336.58,260.03],[0.0578998,0.731559,-0.677849,-0.0446021],
[0,0,-1,5], [-155.949,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_90
:=[[-321.59,133.68,192.85],[0.0858492,0.616175,-0.781151,-0.052549],
[1,0,0,5],[-158.814,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_50
:=[[-278.07,248.67,195.60],[0.138392,0.617445,-0.771524,-0.066035],
[1,0,0,5],[-159.807,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_70
:=[[-440.77,138.00,188.10],[0.0383477,0.528685,-0.847829,-0.0144109],
[1,0,0,5],[-144.248,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_80
:=[[-364.01,303.91,169.25],[0.110077,0.61076,-0.783991,-0.0146054],
[1,1,-1,5],[-153.974,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Pen_100
:=[[-422.88,268.58,181.99],[0.105625,0.564905,-0.815426,-0.0693288],
[1,0,0,5],[-147.315,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
    CONST robtarget Pen_60
    :=[[-489.12,102.46,213.95],[0.359194,0.459281,-0.789068,-0.193424],
    [1,0,0,5],[-137.831,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Rock_10:=[[9.576073371,439.579231074,197.54020098],
    [0.969920392,-0.066595954,-0.231651544,-0.034014329],[0,1,0,4],
    [-82.350193572,9E+09,9E+09,9E+09,9E+09,9E+09]];

    CONST robtarget Board_10
    :=[[-422.04,24.72,150.12],[0.0390186,0.0963675,-0.994102,0.0308575],
    [2,1,1,5],[123.624,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Board_20
    :=[[-418.75,8.25,67.62],[0.0369636,0.0630328,-0.997327,0.000250125],
    [2,1,1,5],[127.717,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Board_30
    :=[[-403.13,9.88,139.82],[0.0456513,0.0855634,-0.994904,0.0275852],
    [2,1,1,5],[123.856,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Board_40
    :=[[-399.79,6.47,140.33],[0.150041,0.0835806,-0.985108,0.00796113],
    [2,0,1,5],[124.157,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Board_50
    :=[[-390.54,110.69,185.12],[0.54843,0.255766,-0.760997,0.233864],
    [1,0,1,5],[-154.194,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !***********************************************************
    !
    ! Procedure main
    !
    !   This is the entry point of your program
    !
    !***********************************************************
    PROC main()
        IF Not InitDone THEN
            !first time running the program
            !initial home position
            g_Init;
            g_Calibrate\Jog;
            WaitTime\InPos,1;
            g_GripOut;
            MoveJ Home_R,v300,z100,tool0\WObj:=RobtR;
            !Interrupts
            IEnable;
            !Enable interrupts
            !create interrupt in case the persistent variable changes value
            CONNECT persCancelint WITH CancelRoutineR;
            IPers MoveCanceled,persCancelint;
            !wait until both arms are in this point
            WaitSyncTask syncTaskDefaultStart,TaskListThree;

        ELSE
```

```
            !when the movement is cancelled, this is run
            connection:=0;
            counter:=0;
            !if it is not the first time the pointer is here because the cancel TRAP has been run
            WaitSyncTask syncTaskFinishCancel1,TaskListThree;
            !wait the TCP/IP task to start movement
            SyncTest;
            WaitSyncTask syncTaskFinishCancel2,TaskListThree;
            !the task has been started
            MoveJ Home_R,v300,z100,tool0\WObj:=RobtR;
            !move the arm to home
            IWatch persCancelint;
            !reactivate interrupts
            WaitSyncTask syncTaskFinishCancel3,TaskListThree;
        ENDIF

        !Forever looop
        WHILE TRUE DO

            !Happy movement
            IF data="good" THEN
                HappyMove;

            ELSEIF data="bad" THEN
                SadMove;

            ELSEIF data="middle" THEN
                SyncTest;
                WaitSyncTask syncTaskReadyMiddle,TaskListDefaultPos;
                IF NOT IsSyncMoveOn() THEN
                    SyncMoveOn syncMoveOnMiddle,TaskListDefaultPos;
                ENDIF
                madR_Path;
                data:="zero";
                SyncTest;
                WaitSyncTask syncTaskFinishMiddle,TaskListDefaultPos;

            ELSEIF data="rock" THEN
                SyncTest;
                WaitSyncTask syncTaskReadyRock,TaskListDefaultPos;
                RockR;
                WaitTime 0.7;
                data:="zero";
                SyncTest;
                WaitSyncTask syncTaskFinishRock,TaskListDefaultPos;

            ELSEIF data="one" THEN
                SyncTest;
                ! ISleep persCancelint;
```

```
                !wait until the other arm has finished the movement
                WaitSyncTask syncTaskReadyOne,TaskListDefaultPos;
                PenPath;
                !grippers path
                WaitUntil data="ready";
                connection:=1;
                !grippers path
                !activate trigger for the other arm
                WaitSyncTask syncTaskFinishOne,TaskListThree;
                connection:=0;
                !wait both programs to be in the same point
                !IWatch persCancelint;

            ELSEIF data="two" THEN
                SyncTest;
                ISleep persCancelint;
                WaitSyncTask syncTaskTwo,TaskListDefaultPos;
                IF NOT IsSyncMoveOn() THEN
                    SyncMoveOn syncMoveOnTaskTwo,TaskListDefaultPos;
                    !synchronize movements
                ENDIF
                BoardPath;
                SyncTest;
                IWatch persCancelint;
                WaitUntil data="ready";
                counter:=0;
                connection:=2;
                WHILE (NumDice<>5) and (counter<>3) DO
                    WaitUntil data="ready";
                    data:="zero";
                    WaitSyncTask syncTaskTwoDice,TaskListTask2;
                    counter:=counter+1;
                ENDWHILE
                !wait until the other arm has finished the movement
                WaitSyncTask syncTaskReadytwo,TaskListDefaultPos;
                IF NumDice=5 THEN
                    HappyMove;
                ELSE
                    SadMove;
                ENDIF
                WaitSyncTask syncTaskFinishtwo,TaskListDefaultPos;
                data:="zero";
                NumDice:= 0;
                counter:=0;
                connection:=0;
                !wait both programs to be in the same point
            ENDIF
        ENDWHILE

    ENDPROC
```

```
    PROC SadMove()
        SyncTest;
        WaitSyncTask syncTaskReadyBad,TaskListDefaultPos;
        IF NOT IsSyncMoveOn() THEN
            SyncMoveOn syncMoveOnBad,TaskListDefaultPos;
        ENDIF
        SadR;
        SyncTest;
        WaitSyncTask syncTaskFinishBad,TaskListDefaultPos;
    ENDPROC

    PROC HappyMove()
        SyncTest;
        WaitSyncTask syncTaskReadyGood,TaskListDefaultPos;
        !wait the other arm
        IF NOT IsSyncMoveOn() THEN
            SyncMoveOn syncMoveOnGood,TaskListDefaultPos;
            !synchronize movements
        ENDIF
        HappyR;
        !happy path of the right arm
        SyncTest;
        WaitSyncTask syncTaskFinishGood,TaskListDefaultPos;
    ENDPROC

    PROC BoardPath()
        ! StartMove;
        MoveJ Home_R\ID:=32,v300,z10,tool0\WObj:=RobtR;
        MoveJ Board_50\ID:=33,v300,z10,tool0\WObj:=RobtR;
        MoveJ Board_10\ID:=23,v300,z10,tool0\WObj:=RobtR;
        MotionSup \Off;
        MoveL Board_20\ID:=24,v300,z10,tool0\WObj:=RobtR;
        WaitTime\InPos,1;
        g_GripIn\holdForce:=15;
        WaitTime\InPos,1;
        MoveL Board_10\ID:=25,v300,z10,tool0\WObj:=RobtR;
        MoveJ Board_30\ID:=26,v100,z10,tool0\WObj:=RobtR;
        MoveJ Board_40\ID:=27,v100,z10,tool0\WObj:=RobtR;
        MoveJ Board_30\ID:=35,v100,z10,tool0\WObj:=RobtR;
        MoveJ Board_40\ID:=36,v100,z10,tool0\WObj:=RobtR;
        MoveJ Board_10\ID:=28,v200,z10,tool0\WObj:=RobtR;
        MoveL Board_20\ID:=29,v100,z10,tool0\WObj:=RobtR;
        WaitTime\InPos,1;
        g_GripOut;
        MoveL Board_10\ID:=30,v300,z10,tool0\WObj:=RobtR;
        MoveJ Board_50\ID:=34,v300,z10,tool0\WObj:=RobtR;
        MoveJ Home_R\ID:=31,v300,z10,tool0\WObj:=RobtR;
    ENDPROC
```

```
    PROC HappyR()
        MoveJ Home_R\ID:=0,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_10\ID:=1,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_20\ID:=2,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_30\ID:=3,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_20\ID:=4,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_10\ID:=5,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_20\ID:=6,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_30\ID:=7,v300,z10,tool0\WObj:=RobtR;
        MoveJ HappyR_20\ID:=8,v300,z10,tool0\WObj:=RobtR;
        MoveJ Home_R\ID:=9,v300,z10,tool0\WObj:=RobtR;
    ENDPROC

    PROC SadR()
        MoveJ Home_R\ID:=10,v500,z10,tool0\WObj:=RobtR;
        MoveJ SadR_10\ID:=11,v500,z10,tool0\WObj:=RobtR;
        MoveJ SadR_20\ID:=12,v200,z10,tool0\WObj:=RobtR;
        MoveJ SadR_10\ID:=13,v200,z10,tool0\WObj:=RobtR;
        MoveJ Home_R\ID:=14,v600,z10,tool0\WObj:=RobtR;
    ENDPROC

    PROC madR_Path()
        MoveJ Home_R\ID:=15,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_10\ID:=16,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_20\ID:=17,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_30\ID:=18,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_20\ID:=19,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_30\ID:=20,v400,z10,tool0\WObj:=RobtR;
        MoveJ madR_20\ID:=21,v400,z10,tool0\WObj:=RobtR;
        MoveJ Home_R\ID:=22,v400,z10,tool0\WObj:=RobtR;
    ENDPROC

    PROC PenPath()
        MoveJ Home_R,v400,z10,tool0\WObj:=RobtR;!move home
        g_GripOut;!open gripper
        MoveJ Pen_10,v400,z10,tool0\WObj:=RobtR;!position over the pen
        MoveL Pen_30,v400,z10,tool0\WObj:=RobtR;!go down, so that the pen is
    in between the gripper fingers
        MotionSup \Off;!diabilitated the collision check to avoid problems
    when the pen is taken
        WaitTime\InPos,1;!wait until the previous function is finished
        g_GripIn\holdForce:=20;!close gripper with a force of 20N
        WaitTime\InPos,1;!wait until it is closed
        MoveL Pen_40,v300,z10,tool0\WObj:=RobtR;!move the pen up
        !make a movement over the industrial piece
        MoveJ Pen_50,v300,z100,tool0\WObj:=RobtR;
        MoveJ Pen_70,v100,z10,tool0\WObj:=RobtR;
        MoveJ Pen_80,v100,z10,tool0\WObj:=RobtR;
        MoveJ Pen_90,v100,z10,tool0\WObj:=RobtR;
        MoveJ Pen_100,v100,z10,tool0\WObj:=RobtR;
```

```
        MoveJ Pen_60,v100,z100,tool0\WObj:=RobtR;!give the pen to the user
        WaitTime\InPos,1;
        nRightHandPos:=g_GetPos();!get the position of the gripper
        nRightHandState:= g_GetState();!get the state of the gripper
        WHILE nRightHandPos<>0 DO!if the gripper is not in position 0 (it is not totally closed)
            nRightHandPos:=g_GetPos();!look the position again
        ENDWHILE
        MoveJ Home_R,v400,z10,tool0\WObj:=RobtR;!move the robot arm to the home position
        g_GripOut;!open the gripper
        g_Stop;!stop the gripper
        MotionSup \On;!enable the collision check
    ENDPROC

    PROC RockR()
        MoveJ Home_R,v300,z10,tool0\WObj:=RobtR;
        MoveJ Rock_10,v300,z10,tool0\WObj:=RobtR;
        WaitTime\InPos,1;
        WaitSyncTask syncTaskRockStop,TaskListDefaultPos;
        MoveJ Home_R,v300,z10,tool0\WObj:=RobtR;
    ENDPROC

    PROC SyncTest()
        IF IsSyncMoveOn() THEN
            WaitSyncTask syncTaskCancel,TaskListDefaultPos;
            SyncMoveOff syncCancel;
        ELSE
            ! Connected Motion Task is in independent mode
        ENDIF
    ENDPROC


    ! In case the cancel gesture is read, this trap/routine will be executed
    TRAP CancelRoutineR
        !diactivate both interrupts
        ISleep persCancelint;
        ExitCycle; !start from main
    ENDTRAP
ENDMODULE
```

B.3 Left robot arm program Code

B.3.1 Data Module

```
MODULE DataModule
```

```

    !*******************************************************
    !Shared data RobL-RobR-TCP_IP_Input
    !*******************************************************
    !interrupts
    PERS bool InitDone;
    PERS bool MoveCanceled;
    PERS string data;
    PERS intnum connection:=0;
    PERS intnum NumDice;

    !*******************************************************
    ! Rob_L data
    !*******************************************************
    PERS tooldata Servo:=[TRUE,[[0,0,114.2],[1,0,0,0]],
    [0.215,[8.7,12.3,49.2],[1,0,0,0],0.00021,0.00024,0.00009]];
    TASK PERS wobjdata RobtR:=[FALSE,TRUE,""
    ,[[0,0,0],[0,0,0,1]],[[0,0,0],[1,0,0,0]]];

     ! T_ROB_L points
    CONST robtarget Home_R:=[[9.578260588,182.610106295,198.627859614],
    [0.523068577,0.111214992,-0.842420958,0.066010745],
    [0,0,0,4],[-101.964436053,9E+09,9E+09,9E+09,9E+09,9E+09]];
  TASK PERS wobjdata RobLWobj:=[FALSE,TRUE,""
    ,[[0,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];

ENDMODULE
```

B.3.2 Main Module

```
MODULE MainModule
    !*********************************************************
    !
    ! Module:  Main Module of the left robot arm
    !
    ! Description:
    !   The program of the lef robot arm for FMT project called: Real-Time
    Collaborative Robot Control Using Hand Gestures, Recognised By Deep
    Learning
    !
    ! Author: leire amenabar
    !
    ! Version: 1.0
    !
    !*********************************************************

    ! Create the task lists
```

```
PERS tasks TaskListDefaultPos{2}:=[["T_ROB_L"],["T_ROB_R"]];
PERS tasks TaskListThree{3}:=[["T_ROB_L"],["T_ROB_R"],["TCP_IP_Input"]];

!wait sync ident
VAR syncident syncTaskDefaultStart;


VAR syncident syncTaskReadyGood;
VAR syncident syncTaskReadyBad;
VAR syncident syncTaskReadyOne;
VAR syncident syncTaskReadytwo;
VAR syncident syncTaskReadyMiddle;
VAR syncident syncTaskReadyRock;

VAR syncident syncTaskFinishGood;
VAR syncident syncTaskFinishBad;
VAR syncident syncTaskFinishOne;
VAR syncident syncTaskFinishtwo;
VAR syncident syncTaskFinishMiddle;
VAR syncident syncTaskFinishRock;

VAR syncident syncTaskFinishCancel1;
VAR syncident syncTaskFinishCancel2;
VAR syncident syncTaskFinishCancel3;
VAR syncident syncTaskCancel;

VAR syncident syncTaskRockStop;

VAR syncident syncTaskTwo;

!synchro making ident
VAR syncident syncMoveOnGood;
VAR syncident syncMoveOnBad;
VAR syncident syncMoveOnStop;
VAR syncident syncMoveOnMiddle;

VAR syncident syncMoveCancel;
VAR syncident syncCancel;

VAR syncident syncMoveOnTaskTwo;


!Interrupts
VAR intnum persStopIntL;
VAR intnum persCancelintL;

!RobotTargets
CONST robtarget target_10:=[[352.995701338,-300.327322519,-85.772716956],
[0.441092614,0.636529641,-0.302462263,-0.555683275],[0,0,0,4],
[101.964426493,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

```
CONST robtarget Target_20:=[[167.036792116,220.524201729,-408.514427477],
[0.879732391,-0.047109949,-0.023568525,-0.472542165],[1,2,-2,0],
[-164.859205264,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget Home_Left:=[[-9.578260574,182.610106314,198.627859608],
[0.066010745,0.842420958,-0.111214992,0.523068577],[0,0,0,4],
[101.964436056,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget HappyL_10:=[[60.958820992,445.606762518,456.448681711],
[0.071967175,0.070401944,-0.088761109,-0.990951945],[-1,0,-1,4],
[-52.90711226,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget HappyL_20:=[[60.956008204,503.645632434,523.61020811],
[0.061023435,0.052121969,0.023208315,-0.996504297],[-1,0,-1,4],
[-112.389624624,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget HappyL_30:=[[60.955858421,503.64542279,672.110468721],
[0.053747604,0.059596788,0.152261375,-0.985076592],[-1,0,-1,4],
[-147.961369864,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget SadL_10:=[[54.945195375,539.030786525,380.398177203],
[0.445042763,-0.481195859,-0.592982284,-0.467717324],[-1,2,-2,4],
[-36.97195114,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget SadL_20:=[[54.945850668,632.63164956,400.458097163],
[0.445043016,-0.481196176,-0.59298196,-0.467717168],[-1,2,-2,4],
[-36.971997946,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget madL_10:=[[407.114313539,182.610106314,198.627859608],
[0.066010745,0.842420958,-0.111214992,0.523068577],[0,0,0,4],
[101.964436056,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget madL_20:=[[407.114432752,182.610260456,198.627848504],
[0.120500657,0.341553057,-0.046966071,0.930921742],[-1,2,-2,4],
[101.964434161,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget madL_30:=[[407.114518113,182.610478826,198.627743895],
[0.128731831,0.075505766,-0.012424552,0.988722724],[-1,0,0,4],
[101.964435427,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget RockL_10:=[[284.693111647,204.957290926,147.913674178],
[0.473127174,0.466525842,-0.631593808,0.399491649],[-1,0,-1,4],
[159.664820838,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget RockL_20:=[[284.693223049,204.957482636,229.150777492],
[0.473127256,0.466525929,-0.631593826,0.399491421],[0,0,-1,4],
[159.664826644,9E+09,9E+09,9E+09,9E+09,9E+09]];

CONST robtarget BoardL_10
:=[[407.85,317.20,141.92],[0.0241279,-0.99954,-0.00782593,-0.016612],
[-1,-1,0,5],[155.537,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget BoardL_20
:=[[407.26,322.93,54.03],[0.0286426,-0.999113,-0.0295573,-0.00892847],
[-1,-1,0,5],[152.43,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget BoardL_30
:=[[390.30,332.21,122.42],[0.0154118,-0.999247,0.0152883,-0.0321523],
```

```
    [-1,-1,0,5],[150.682,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget BoardL_40
    :=[[404.24,327.88,135.95],[0.0308986,-0.993916,0.0214404,-0.103524],
    [-1,-1,0,5],[149.731,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget BoardL_50
    :=[[347.56,324.73,298.26],[0.0275514,-0.962651,0.0500273,-0.264653],
    [-1,-1,0,5],[141.661,9E+09,9E+09,9E+09,9E+09,9E+09]];


    !Main procedure
    PROC main()
        IF Not InitDone THEN
            !variable initialization
            MoveCanceled:=FALSE;
            g_Init;
            g_Calibrate\Jog;
            WaitTime\InPos,1;
            g_GripOut;
            !initial home position
            MoveJ Home_Left,v300,z10,tool0;
            !g_Init \maxSpd := 20, \holdForce := 10;
            !Interrupts
            IEnable;
            !Enable interrupts
            !create interrupt in case the persistent variable changes value
            CONNECT persCancelintL WITH CancelRoutineL;
            IPers MoveCanceled,persCancelintL;
            !wait until both arms are in this point
            WaitSyncTask syncTaskDefaultStart,TaskListThree;

        ELSE
            !when the movement is cancelled, this is run
            connection:=0;
            WaitSyncTask syncTaskFinishCancel1,TaskListThree;
            SyncTest;
            WaitSyncTask syncTaskFinishCancel2,TaskListThree;
            MoveJ Home_Left,v300,z10,tool0;
            !move the arm to home
            IWatch persCancelintL;
            !reactivate interrupts
            WaitSyncTask syncTaskFinishCancel3,TaskListThree;
        ENDIF

        !Forever loop
        WHILE TRUE DO

            IF data="good" THEN
                HappyMoveL;

            ELSEIF data="bad" THEN
```

```
                SadMoveL;

            ELSEIF data="middle" THEN
                SyncTest;
                WaitSyncTask syncTaskReadyMiddle,TaskListDefaultPos;
                IF NOT IsSyncMoveOn() THEN
                    SyncMoveOn syncMoveOnMiddle,TaskListDefaultPos;
                ENDIF
                madL_Path;
                WaitTime 0.7;
                data:="zero";
                SyncTest;
                WaitSyncTask syncTaskFinishMiddle,TaskListDefaultPos;

            ELSEIF data="rock" THEN
                SyncTest;
                WaitSyncTask syncTaskReadyRock,TaskListDefaultPos;
                Rock_L;
                WaitTime 0.7;
                data:="zero";
                SyncTest;
                WaitSyncTask syncTaskFinishRock,TaskListDefaultPos;

            ELSEIF data="one" THEN
                SyncTest;
                !ISleep persCancelintL;
                !diactivate interrupt
                !task one
                WaitSyncTask syncTaskReadyOne,TaskListDefaultPos;
                WaitSyncTask syncTaskFinishOne,TaskListThree;
                data:="zero";
                !IWatch persCancelintL;
                !reactivate the interrupt

            ELSEIF data="two" THEN
                SyncTest;
                ISleep persCancelintL;
                WaitSyncTask syncTaskTwo,TaskListDefaultPos;
                IF NOT IsSyncMoveOn() THEN
                    SyncMoveOn syncMoveOnTaskTwo,TaskListDefaultPos;
                    !synchronize movements
                ENDIF
                BoardPath;
                SyncTest;
                IWatch persCancelintL;
                !wait until the other arm has finished the movement
                WaitSyncTask syncTaskReadytwo,TaskListDefaultPos;
                IF NumDice=5 THEN
                    HappyMoveL;
                ELSE
```

```
                    SadMoveL;
                ENDIF
                !activate trigger for the other arm
                WaitSyncTask syncTaskFinishtwo,TaskListDefaultPos;
                !wait both programs to be in the same point
            ENDIF
        ENDWHILE

    ENDPROC


    PROC SadMoveL()
        SyncTest;
        WaitSyncTask syncTaskReadyBad,TaskListDefaultPos;
        IF NOT IsSyncMoveOn() THEN
            SyncMoveOn syncMoveOnBad,TaskListDefaultPos;
        ENDIF
        !synchronize both arms
        SadL;
        !sad path
        data:="zero";
        !so that it does not repeat the same movement
        SyncTest;
        !every syncmove before put a waitsync
        WaitSyncTask syncTaskFinishBad,TaskListDefaultPos;

    ENDPROC

    PROC HappyMoveL()
        SyncTest;
        WaitSyncTask syncTaskReadyGood,TaskListDefaultPos;
        !wait the other arm
        IF NOT IsSyncMoveOn() THEN
            SyncMoveOn syncMoveOnGood,TaskListDefaultPos;
            !synchronize movements
        ENDIF
        HappyL;
        WaitTime 0.7;
        data:="zero";
        SyncTest;
        WaitSyncTask syncTaskFinishGood,TaskListDefaultPos;
    ENDPROC

    PROC BoardPath()
        MoveJ Home_Left\ID:=32,v300,z10,tool0;
        MoveJ BoardL_50\ID:=33,v300,z10,tool0;
        MoveJ BoardL_10\ID:=23,v300,z10,tool0;
        MotionSup \Off;
        MoveL BoardL_20\ID:=24,v300,z10,tool0;
        WaitTime\InPos,1;
```

```
        g_GripIn\holdForce:=15;
        WaitTime\InPos,1;
        MoveL BoardL_10\ID:=25,v300,z10,tool0;
        MoveJ BoardL_30\ID:=26,v100,z10,tool0;
        MoveJ BoardL_40\ID:=27,v100,z10,tool0;
        MoveJ BoardL_30\ID:=35,v100,z10,tool0;
        MoveJ BoardL_40\ID:=36,v100,z10,tool0;
        MoveJ BoardL_10\ID:=28,v300,z10,tool0;
        MoveL BoardL_20\ID:=29,v300,z10,tool0;
        WaitTime\InPos,1;
        g_GripOut;
        MoveL BoardL_10\ID:=30,v300,z10,tool0;
        MoveJ BoardL_50\ID:=34,v200,z10,tool0;
        MoveJ Home_Left\ID:=31,v100,z10,tool0;

    ENDPROC

    PROC HappyL()
        MoveJ Home_Left\ID:=0,v300,z10,tool0;
        !without anything means that workobject0 is taken as base.
        MoveJ HappyL_10\ID:=1,v300,z10,tool0;
        MoveJ HappyL_20\ID:=2,v300,z10,tool0;
        MoveJ HappyL_30\ID:=3,v300,z10,tool0;
        MoveJ HappyL_20\ID:=4,v300,z10,tool0;
        MoveJ HappyL_10\ID:=5,v300,z10,tool0;
        MoveJ HappyL_20\ID:=6,v300,z10,tool0;
        MoveJ HappyL_30\ID:=7,v300,z10,tool0;
        MoveJ HappyL_20\ID:=8,v300,z10,tool0;
        MoveJ Home_Left\ID:=9,v300,z10,tool0;
    ENDPROC

    PROC SadL()
        MoveJ Home_Left\ID:=10,v500,z10,tool0;
        MoveJ SadL_10\ID:=11,v500,z10,tool0;
        MoveJ SadL_20\ID:=12,v200,z10,tool0;
        MoveJ SadL_10\ID:=13,v200,z10,tool0;
        MoveJ Home_Left\ID:=14,v600,z10,tool0;
    ENDPROC

    PROC madL_Path()
        MoveJ Home_Left\ID:=15,v400,z10,tool0;
        MoveJ madL_10\ID:=16,v400,z10,tool0;
        MoveJ madL_20\ID:=17,v400,z10,tool0;
        MoveJ madL_30\ID:=18,v400,z10,tool0;
        MoveJ madL_20\ID:=19,v400,z10,tool0;
        MoveJ madL_30\ID:=20,v400,z10,tool0;
        MoveJ madL_20\ID:=21,v400,z10,tool0;
        MoveJ Home_Left\ID:=22,v400,z10,tool0;
    ENDPROC

```

```
    PROC Rock_L()
        MoveJ Home_Left,v300,z10,Servo\WObj:=wobj0;
        MoveJ RockL_10,v300,z10,Servo\WObj:=wobj0;
        MoveL RockL_20,v200,z100,Servo\WObj:=wobj0;
        MoveL RockL_10,v200,z100,Servo\WObj:=wobj0;
        MoveL RockL_20,v200,z100,Servo\WObj:=wobj0;
        MoveL RockL_10,v200,z100,Servo\WObj:=wobj0;
        MoveL RockL_20,v200,z100,Servo\WObj:=wobj0;
        MoveL RockL_10,v200,z100,Servo\WObj:=wobj0;
        WaitTime\InPos,1;
        WaitSyncTask syncTaskRockStop,TaskListDefaultPos;
        MoveJ Home_Left,v300,z10,Servo\WObj:=wobj0;
    ENDPROC

    PROC SyncTest()
        !   WaitSyncTask syncTaskCancel,TaskListDefaultPos;
        IF IsSyncMoveOn() THEN
            WaitSyncTask syncTaskCancel,TaskListDefaultPos;
            SyncMoveOff syncCancel;
        ELSE
            ! Connected Motion Task is in independent mode
        ENDIF
    ENDPROC



    ! In case the cancel gesture is read, this trap/routine will be executed
    TRAP CancelRoutineL
        !diactivate the cancel interrupts
        ISleep persCancelintL;
        !start from main
        ExitCycle;
    ENDTRAP


ENDMODULE
```

Appendix C

Implementation

To understand better the implementation of all the parts performed in the project, the trained NN and the robot programs, with EAVS program observe Figure C.1. Here all the process in its totality is explained. The boxes in blue are actions performed by EAVS and the boxes in green are the ones performed by the robot. The ANN would be inserted in EAVS program, EAVS program would run the selected ANN depending in the received message from the robot.

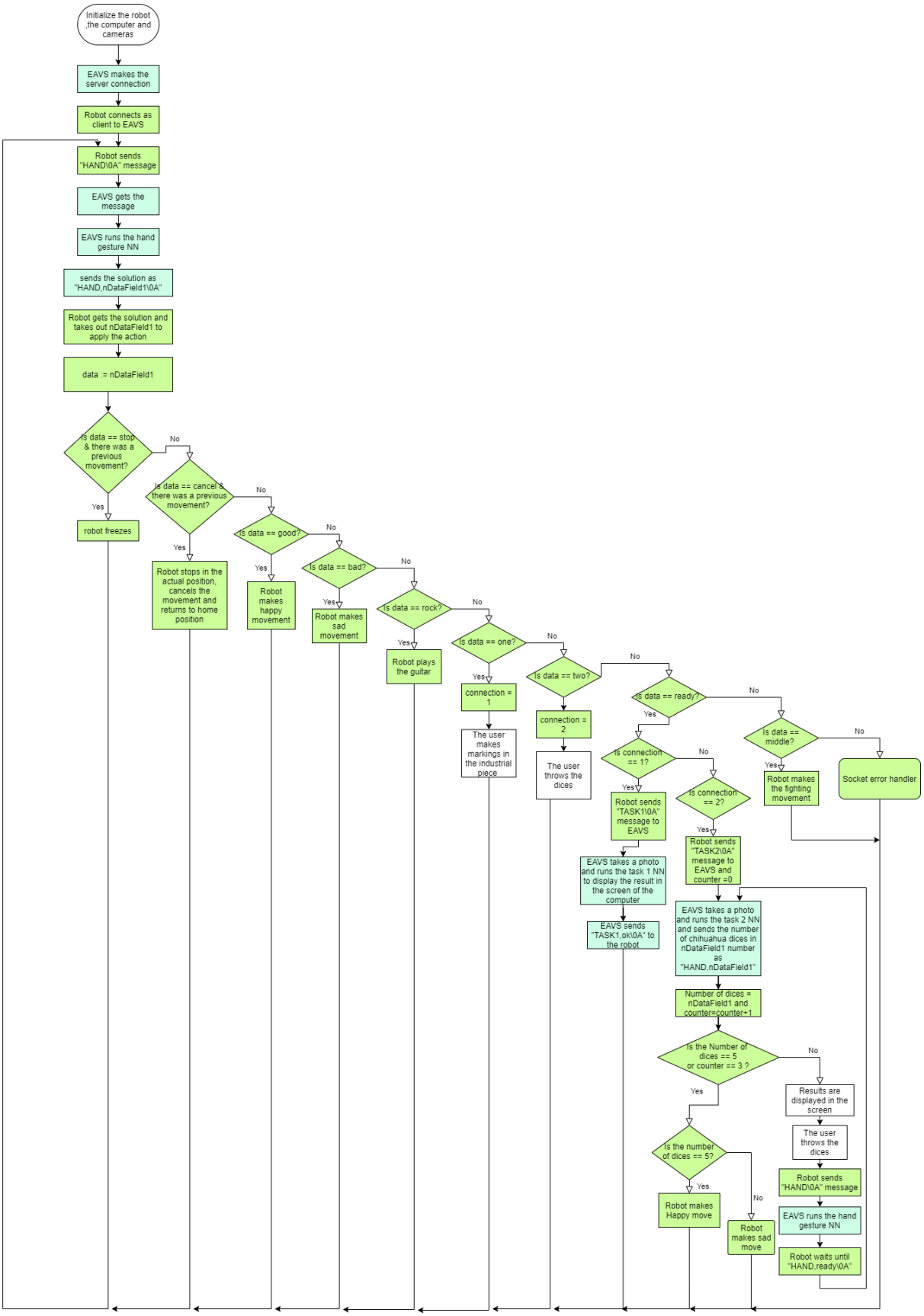


Figure C.1: Final implementation flow chart

Appendix D

Future improvements

D.1 Sound

For a better communication between the user and the robot would be interesting to add sounds to the robot. As the YuMi robot does not have speakers for the reproduction of the sounds, a possible solution would be by the use of Arduino MKRZero which has the possibility to insert an SD card with ".wav" sound archives. As the YuMi robot has digital outputs, this could be connected to the digital inputs of the board to give digital signal to select the sound that is wanted to be played. As the maximum voltage for the digital signals in the board is of 3.3v and the signal given by the robot is of 24v, a voltage divisor would have to be made (see Figure D.1).

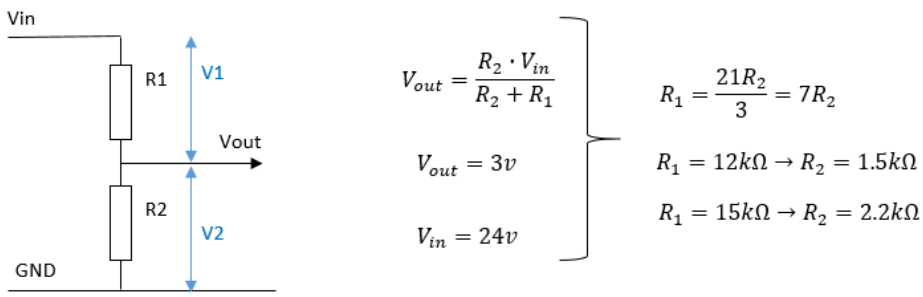


Figure D.1: Voltage Divisor

Apart from the voltage divisor, a sound amplifier is needed. This has been obtained from the web page `https://www.arduino.cc/en/Tutorial/SimpleAudioPlayer` which uses a LM386 microchip and can be observed in the Figure D.2. This circuit would need an external source to aliment the LM386, which could be fixed with a battery of 9v. Apart from that, the Vin signal would have to be connected to the Arduino board as well as the GND. By the connection of all GND external noises are partially avoided.

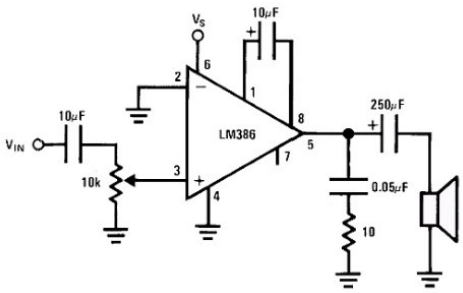


Figure D.2: Sound amplifier

D.2 Facial expressions (UI)

For a more interactive communication between the user and the robot, an screen with facial expressions could be added to the robot. To get ideas of which would be proper facial expressions for the robot, the [48] and [49] articles have been read. From them, the facial expressions represented in Figure D.3 have been designed.

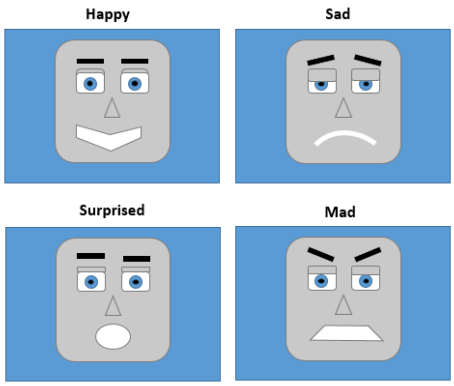


Figure D.3: Robot facial expressions

As can be seen in the Figure D.3, there are four facial expressions, *happy*, *sad*, *surprised* and *mad*. This facial expressions would be linked to the emotional movements of the robot. The *happy* face will go with the victory or *happy* movement, the *sad* face will be linked to the failure or *sad* movement. When it comes to the *surprised* face, this will be used when someone asks the robot to play the guitar, that is to say when the user makes the *rock* gesture. To finish, the *mad* face will be used when a user shows the *middle* finger and the robot answers with the *mad* movement.