Motion capture methods and machine learning for sign language recognition

Norges teknisk-naturvitenskapelige universitet

Jorge Casas Guerrero
Jorge Cristobal Martin
Javier Luengo Peñas

3rd of May, 2019

Abstract

Communication is one of the key elements of all humans. There are a variety of possible ways to communicate, such as written, speech or sign language. The latter is one of the most important since it is used by people who is not able to communicate through speech. Most motion capture studies had try to recognize sign language in order to teach it. In most of them, technologies such as marker gloves or expensive cameras has been used to achieve the recognition. In this project, the recognition of sign language using affordable and optimal technologies has been studied, so most of the people can access them. In this way, a comparison of several technologies and the development and evaluation of a software system using the chosen one is carried out, building a final system using the Leap Motion Controller, Microsoft Kinect and TensorFlow, which is capable of recognizing 17 static sings, with an accuracy of almost 100%, and dynamic signs.

Resumen

La comunicación es uno de los elementos clave de todos los seres humanos. Hay una variedad de formas posibles de comunicarse, como el lenguaje escrito, la comunicación verbal o el lenguaje de señas. Este último es uno de los más importantes, ya que es utilizado por personas que no son capaces de comunicarse a través del habla. La mayoría de los estudios de captura de movimiento han tratado de reconocer el lenguaje de signos para poder enseñarlo. En la mayoría de ellos se han utilizado tecnologías como los guantes con marcadores o las costosas cámaras para captar el reconocimiento. En este proyecto se ha estudiado el reconocimiento del lenguaje de signos mediante tecnologías asequibles y óptimas, para que la mayoría de la gente pueda acceder a ellas. De esta manera, se realiza una comparación de varias tecnologías y el desarrollo y evaluación de un sistema de software utilizando la tecnología elegida, construyendo un sistema final utilizando el Leap Motion Controller, Microsoft Kinect y TensorFlow, que es capaz de reconocer 17 signos estáticos, con una precisión de casi el 100 %, y signos dinámicos.

Acknowledgements

We would like to thank Tomas and Alexander Holt and the NTNU for giving us the opportunity of doing our bachelor thesis during this semester.
We are grateful to Vangos Pterneas for his tutorials and open source codes for programming with the Kinect V2.
Also we would like to thank our families for giving us the chance to come to study to Norway and supporting us during this year.

Contents

Introduction

Communication is one of the key elements of all humans. It's one of the abilities that define us and helped us evolve. To establish a communication, two persons are needed. This two persons need to know the same concepts and rules in order to understand each other. These concepts and rules are what is called language. Oral communication is the most common communication used, where the transmitter person uses speech and the receiver uses hearing. Some people are born without the ability of speak or to hear, or others lose them later by other means. These people communicate through sign language, in which visual movements of the body and hands are use to convey meaning.

In many countries, people who speaks sign language cannot afford the attendance at sign language lessons due to commuting, cost, time, etc. Other people may want to learn it in order to communicate with them. Today's alternatives are technologies and devices such as hearing aids, implants or assistive devices, but they are expensive and may not be the solution to most of the people. Other solution could be a translator between a hearing impaired person and a normal hearing person. With today's technologies, this translator could be a computer or electronic device.

Therefore, one alternative is to create a software system which could teach and translate sign language through an AI to a user. This system would require to be as cheap as possible as well as to be very precise, since sign language involves finger positioning as well as facial expressions. It would read the signs from the hearing impaired person and translated it. So, a person who is willing to study the sign language could read the letter the AI is asking for and perform the sign at task, receiving a feedback about if the sign it's performed well, or it could translate the signs for people that don't understand sign language, getting the role of translator.

In this project, the recognition of sign language using affordable and optimal technologies, such as Microsoft Kinect or Leap Motion, has been studied, so most of the people can access it. In this way, a comparison of this technologies and the development and evaluation of a software system using the chosen one is carried out.

Theory

2.1 Sign Language

Sign language is a natural language of expression and gesture-spatial configuration and visual perception [1], through which deaf people can communicate with their social environment, both with other hearing-impaired people and with other individuals who know the sign language used. Sign language is a language with its own grammatical rules and sentence structure [2].
Although there is an international alphabet, different countries have their own national sign language. One-hand alphabets are used primarily to express personal names and/or place names that have characters of the language used or when that word cannot be expressed with a single gesture. The spelling of the language of a hand can be used to express foreign words, uncommon terms...
The Norwegian Association of the Deaf has made its own handwritten alphabet in collaboration with the Extrast Foundation, a file with several variations of letters and a file with a single variant of letters. This is because different letter characters and "international" character characters have become relatively common among sign language users in Norway, so they decided to create two files [3].

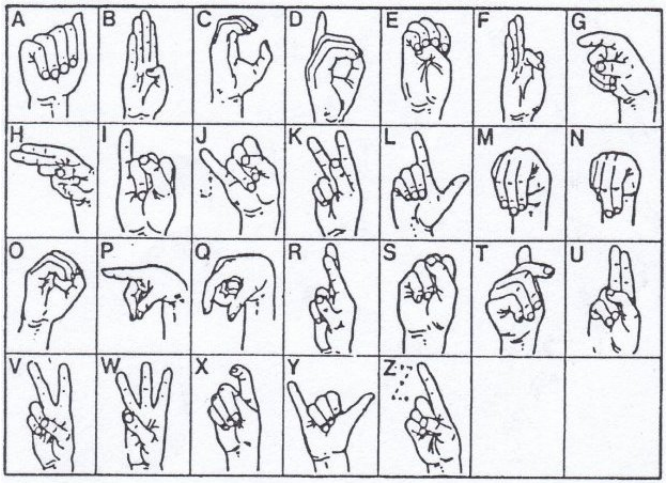


Figure 2.1: International Sign Language alphabet

2.2 Mathematical algorithms

2.2.1 The euclidean distance

The euclidean distance algorithm calculates the distance between two points in a Euclidean space, also known as Cartesian space or simply n-space . An Euclidean space is the space of all n-tuples of real numbers, $(x_1, x_2, x_3, \cdots, x_n)$ [4]. Such n-tuples are also called points. The Euclidean distance is computed using the following formula [5]:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}$$

Where p and q are the points and n the number of dimensions.

2.2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm used to compare the similarity of two sequences which may vary in time or speed [6]. The sequences can be placed on both sides of a grid, being the origin the bottom left corner. Each cell will have a measured distance between the elements of the two sequences. The best match would be the resulting path in which the minimum distance is obtained between the alignment of the sequences.

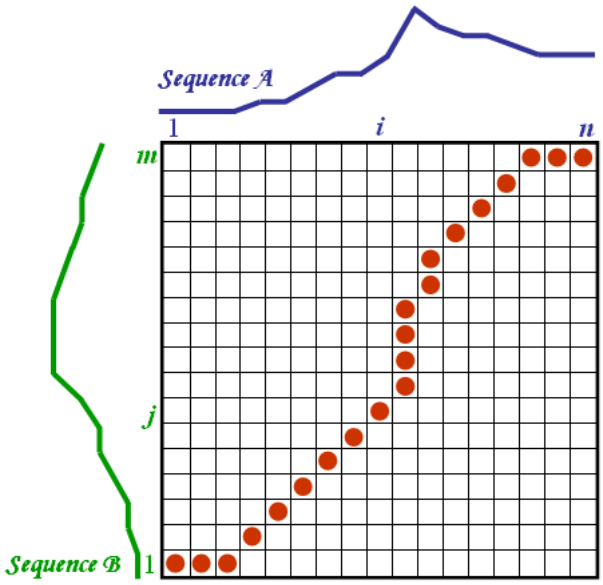


Figure 2.2: Dynamic Time Warping Grid [7]

2.2.3 Cosine Similarity

Cosine similarity is a metric used to determine how similar two vectors are, regardless of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multidimensional space [8]. This function provides a value between an interval between -1 and 1 ([-1,1]). Being 1 if the angle is zero, meaning that both vectors point to the same place, being 0 if the vectors were orthogonal, cancelling out

the cosine in this way, and being -1 if the vectors point in opposite directions. This function is usually used in search and retrieval of information represented values in a vector space.
The function would be the following one [8]:

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{2.1}$$

2.3 Motion Capture

According to the book Motion Capture Performance (2015) we define Motion Capture, as known as MoCap, as a series of techniques where actors wear specially designed suits allowing computers to track their movements. These captured movements are used as the basis for lifelike animated characterization as we can see in the Figure 2.3 [9].
The movements of the actors are recorded from several cameras several times per second (depending on the camera), in order to calculate the 3D position of these and then be used in animations.
This recording method offers advantages over traditional computer animation:

- Low latency, similar to real time
- Complex movements with greater reality, and more realistic physics.
- Greater productivity in less time, therefore lower costs.
- The only limitation, the performance of the actor

Figure 2.3: MoCap of Gollum (*The lord of the rings*)

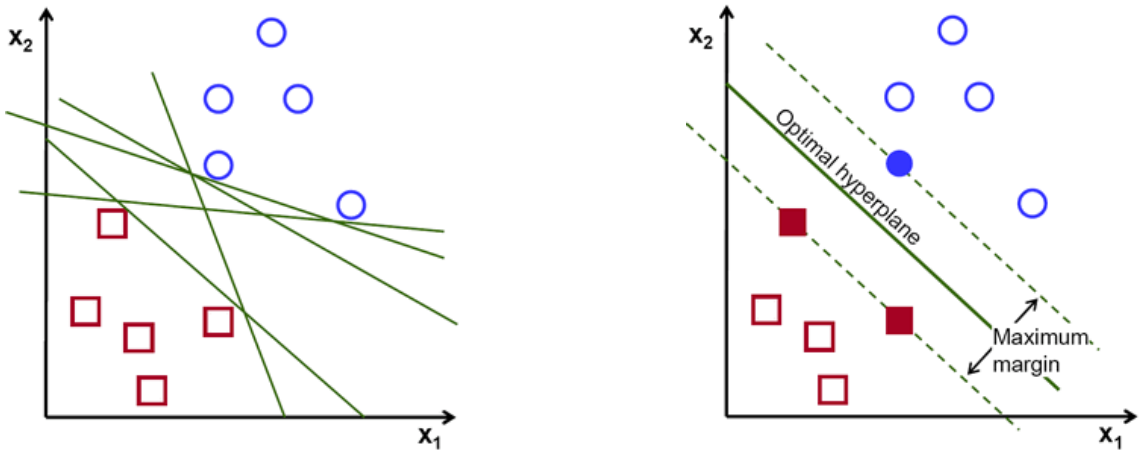
2.4 Machine Learning

Machine Learning is a branch of artificial intelligence (AI) that using the application of algorithms, provides systems the ability to learn and improve automatically from experience without being explicitly programmed. The computational characteristic of Machine Learning is to estimate a target function returning an hypothesis by generalizing the training experience. The goal of Machine Learning is to predict future scenarios that are unknown to the computer based on its previous knowledge.

There are several methods of Machine Learning implementation. In the project described in this report two of them have been taking into account, Support Vector Machine (SVM) and Artificial Neural Networks (ANN). According to the aim of this project, only a Neural Network have been created.

2.4.1 Support Vector Machine

Support Vector Machine is a supervised Machine Learning algorithm commonly used for in classification scenarios, but it can also solve regression problems. A Support Vector Machine performs a classification by calculating an hyperplane, in an N-dimensional space, that separates both classes, maximizing the margin between them.



(a) Possible Hyperplanes

(b) Optimal Hyperplane

Figure 2.4: Comparison of hyperplanes in *SVM* [10]

There is an infinite number of hyperplanes that divides the two classes, as seen in Fig. 2.4, but there is only one hyperplane that maximizes the distance between data points of both classes, providing enough reinforcement so that future data points can be classified with more accuracy. This optimal hyperplane can be seen in Fig. 2.4, and so the maximum margin between classes.

2.4.2 Artificial Neural Networks

Neural Networks is a Machine Learning technique that, as it can be assumed by its name, emulate the architecture of the biological neural systems. Neural Networks

are part of the sub-symbiotic paradigm, as it simulates the fundamental physical (neural) processes in the brain [11]. Neural Networks are formed by learning units called neurons, as can be seen in Fig. 2.5, that convert an input (samples of data) into an output (labels or classes), by learning relations *input-output* through recognizing a huge amount of samples.

Biological Fundamentals

According to biology, a neuron is the fundamental unit of the nervous system. The basic purpose of a neuron is to receive incoming information and, based upon that information, send a signal to other neurons, muscles, or glands [12]. Neurons receive information by stimuli through the dendrites and the information signal is sent between them through the axon.

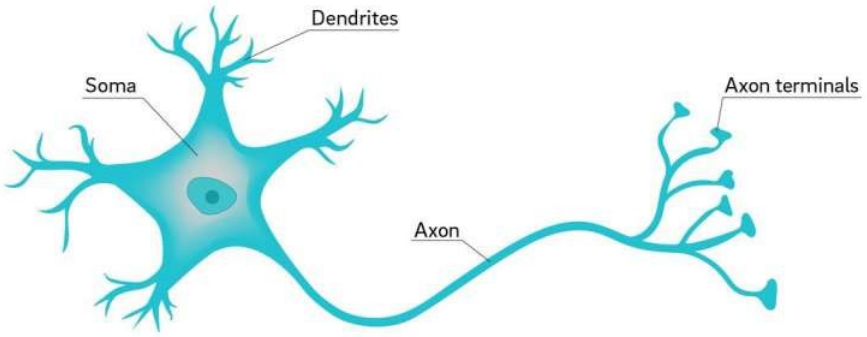


Figure 2.5: Structure of a Neuron

The function of a neuron within a neural system is to receive information, decode that information and send it again to another neuron. Neurons are not linear elements. The nervous system works as a net that propagates electrochemical signals from cell to cell, modifying the ion concentration of the synapses. They are capable of manage inconsistent fuzzy and noisy information due to their sturdiness and adaptability. The human brain contains 100 billion neurons and 10 times more glial cells[13].

Artificial Neuron

Each neuron is unique within a neural network. This means that each one is characterized by an internal state denominated activation level. This activation level has two different values activated or deactivated. Each neuron has a *weight vector* $(w_1, w_2, \cdots, x_n)$ [14], where n is the number of inputs to that neuron. The weights of each neuron are tuned during the training stage. All those input signals combines together to generate the total input. This total input is generated by the summation function [14]:

$$f(x_1, x_2, \cdots, x_n) = x_1 * w_1 + x_2 * w_2 + \cdots + x_n * w_n$$

The non-linear behaviour within a neural network is accomplished by the use of an activation function. The activation function of the neuron changes the activation level through the input signals. In this project two different activation functions have been used:

- ***ReLU:***ReLU stands for Rectified Linear Unit. Mathematically, it is defined as $y = max(0, x)$, Fig. 2.6. ReLU is linear for all positive values, and zero for all negative ones. Regarding computation time, this function is not complicated, so it take less time to train or run.

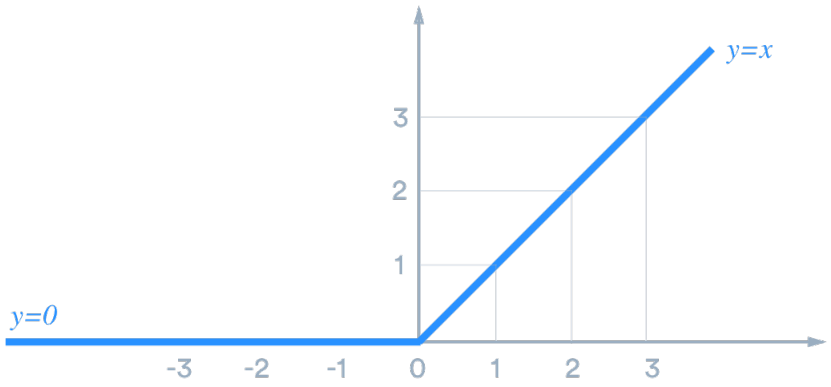


Figure 2.6: Mathematical representation of ReLU [15]

- ***Softmax:*** This function is used in the output layer because it turns numbers into probabilities that sum to one. Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes. The output of the net is the probability of an input vector to belong to a class, Fig. 2.7.

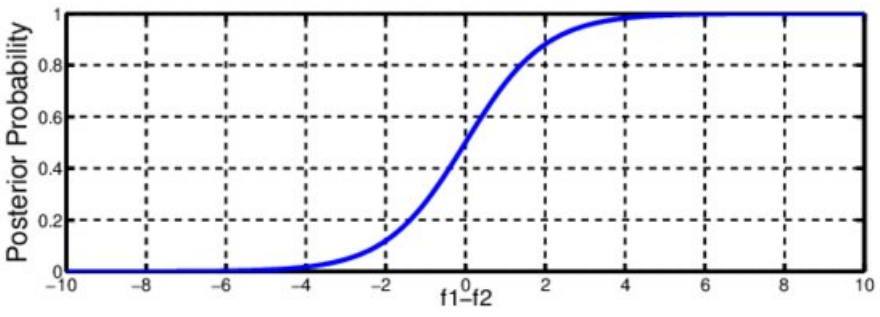


Figure 2.7: Mathematical representation of Softmax [16]

Artificial Neural Network

A neural network is formed by different layers and each one of them contains a certain number of artificial neurons that will take an input and provide an output.

All these inputs will be processed for the activation functions of the neurons within a layer and will produce an output that will be passed as input to the next layer. Every neural network has at least one input layer, that receives the raw data set, and one output layer, will be in charge of return the final calculation of the network. Between those two layers can be found what is called hidden layers. This name is because the value of this layers is not observed in the training stage, and each one of them (if there are more than 1 hidden layer) can have different activation functions.

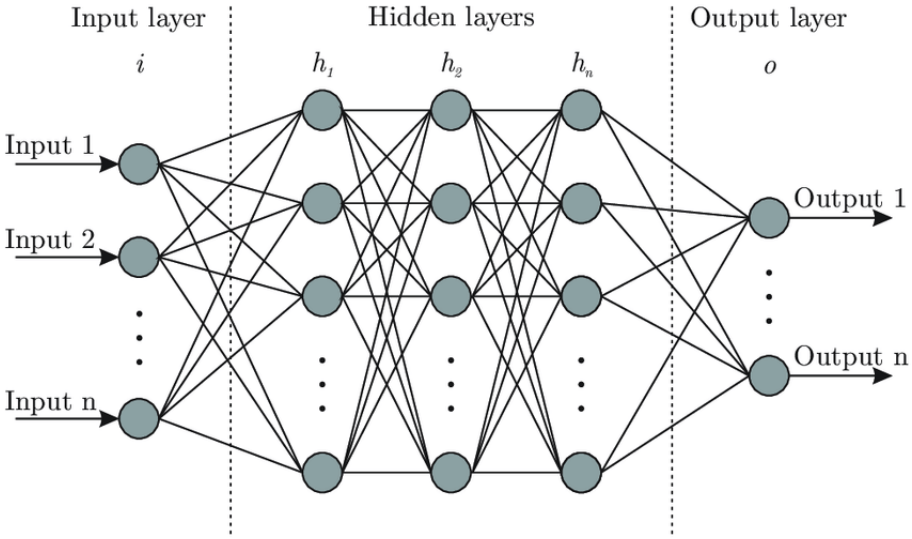


Figure 2.8: Architecture of a Neural Network [17]

Once the architecture of the network is defined, is time for the training stage. The very first step of this stage, is to pre-process the data that the network is going to use for learning. The data has to be randomized and normalized according to the following formula:

$$\mathbf{X}' = \frac{\mathbf{X} - \mathbf{X_{min}}}{\mathbf{X_{max}} - \mathbf{X_{min}}} \tag{2.2}$$

When the data set is randomized and normalized, it is usual to divide it in three sub-sets. 70% of the data will be used for training the network, 15% for validation and the other 15% for testing.

The training stage consists in minimize a loss function. This loss function depends of the problem that the net is going to solve. In this project the net is facing a classification problem, so the loss function is *binary cross-entropy* [18].

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1)) \tag{2.3}$$

The loss can be expressed as [18]:

$$CE = \begin{cases} -log(f(s_1)) & if \quad t_1 = 1 \\ -log(1 - f(s_1)) & if \quad t_1 = 0 \end{cases}$$

It is used for multi-label classification because the loss computed for every neural network output vector component is not affected by other component values.

There is a common problem in machine learning systems called overfitting. Overfitting is when a neural network achieve a really high accuracy with the training set and it get used to it, so in the test phase the network does not works properly under new data that it has never seen before. In order to prevent this a dropout between the layers of the network is applied. This dropout consist in disconnect a percentage of the neurons in each iteration in the training phase.

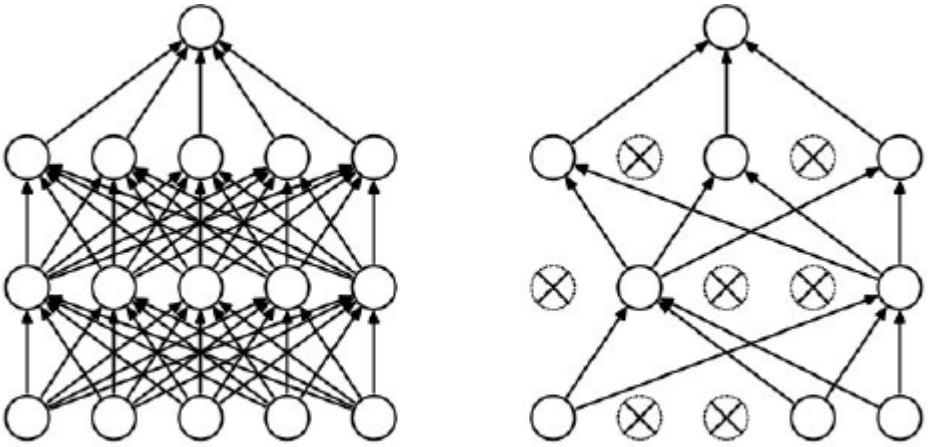


Figure 2.9: Droput schema [14]

2.4.3 Hidden Markov Model

Hidden Markov Model (HMM) is a statistical Markov model. It represent probability distributions over sequence of observations [19]. It has two important properties: An observation, data we know and can observe, at time t produced by a process has a state which is hidden from the observer, and that this state satisfies the Markov property: at a given time of an state, the future of the process only depends of the present state and not from previous events of the process.
The next image, Fig 2.10, shows Markov processes between the Hidden Layers, which are *Rainy* and *Sunny*. *Walk, Clean* and *Shop* are the Observation of the model.

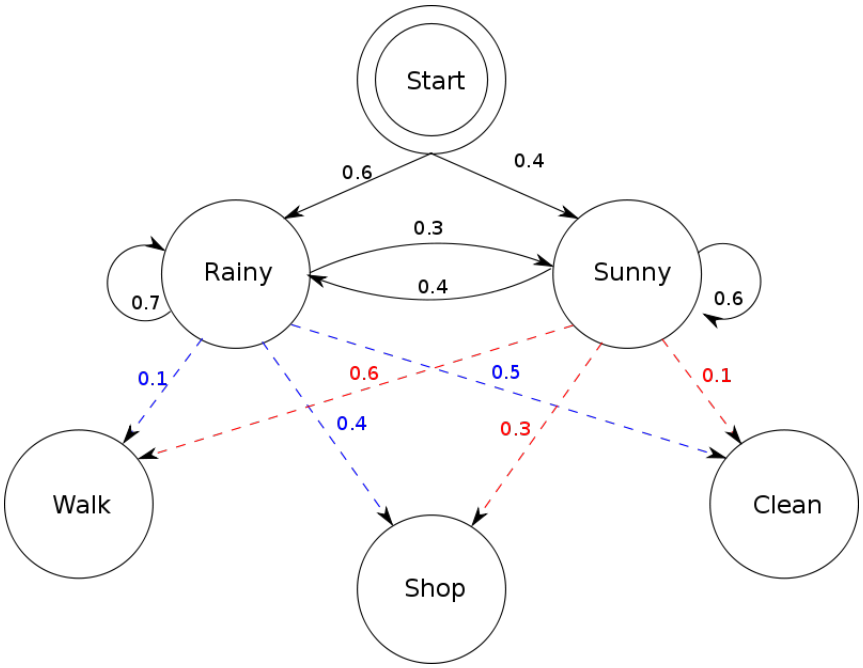


Figure 2.10: Hidden Markov Model example [20]

The black arrows connecting the *Start* with the Hidden Layers are the state transition probabilities, which are the probabilities of staying in the same state or change to another one. The probability in which the model start by going to an initial state is called *Initial State Distribution*. The red and blue arrows are the Observation probability matrix, given a state, the probability of going to one Observation or to another. The probability of an Observation is given by multiplying the initial state distribution and emission probability matrix.

Technology and Methodology

3.1 Leap Motion Controller

3.1.1 Introduction

The Leap Motion Controller is a small USB peripheral that recognizes and tracks your hands and fingers. It is designed to be placed on a desk, facing up as shown in the Fig 3.11, or mounted on virtual reality cameras. The device works at close range with high precision and high frame rate tracking and returns hand and finger positions.

Figure 3.11: The Leap Motion controller's view of your hands [21]

The Leap Motion Controller consists of two monochromatic IR cameras and three infrared LEDs these sensors are directed along the axis-and achieving a field of view of approximately 150 degrees when the device is in its standard operating position. The effective range of the Leap Motion Controller extends from 25 millimetres to approximately 600 millimetres, which is not a very wide range for movements that will cover a large space.
The controller's detection and tracking is most efficient when it has a clear view and high contrast of the object's shape. Its software combines data from its sensors with an internal model of the human hand to facilitate monitoring when conditions are not optimal.

3.1.2 Coordinate system

It uses a Cartesian coordinate system for right-handed people. The origin of this system is in the center of the upper part of the device. The X and Z axes are in the horizontal plane, the X axis being parallel to the long edge of the Leap Motion Controller and the Z axis perpendicular to it. The Y-axis is vertical, with positive

values increasing upward (unlike most computer graphics coordinate systems which have a downward orientation) Fig 3.12. The Z-axis values increase as they approach the user.

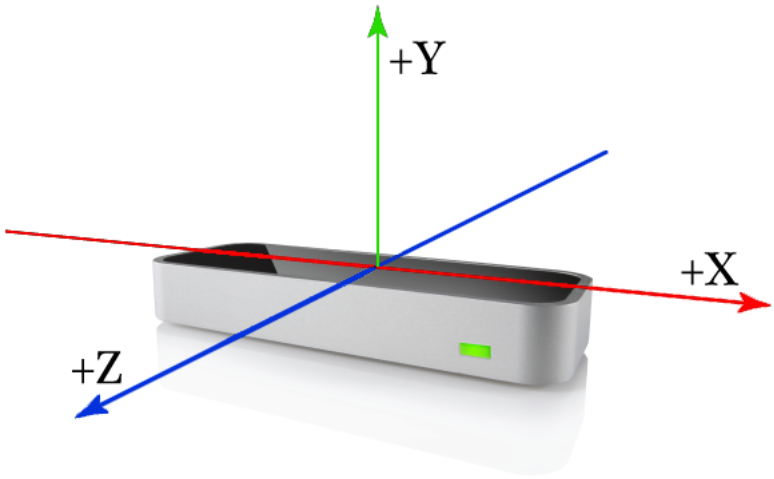


Figure 3.12: The Leap Motion right-handed coordinate system [21]

The Leap Motion API measures physical quantities with the following units[21]:

Physical quantities	Units
Distance	millimeters
Time	microseconds (unless otherwise noted)
Speed	millimeters/second
Angle	radians

3.1.3 Motion Tracking Data

As the Leap Motion controller tracks hands and fingers it provides updates as a set (or frame) of data. Each Frame object contains any hand tracked, detailing its properties at a specific point in time. The Frame object is the "root" of the Leap Motion data model.

Hands:

The hand model provides information about identity, position and other features such as the arm to which the hand is attached and the lists of fingers associated with the hand.
As mentioned earlier, the Leap Motion software uses an internal model of a human hand to predict the movement of the hand when parts of a hand are not visible. The hand model provides positions for its five fingers. Subtle movements of fingers inside the hand or in positions unreachable by sensors are often not detected, Fig 3.13. More than two hands can be detected in the same frame. However it is advisable to keep at most two hands in the field of vision for optimal quality of movement tracking.

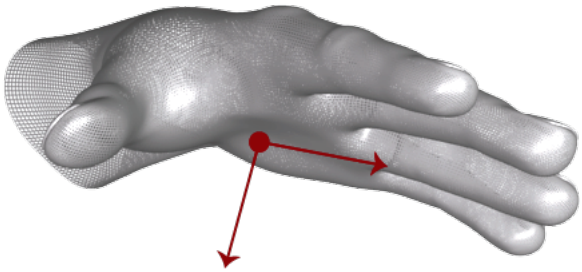


Figure 3.13: The hand palm normal and direction vectors define the orientation of the hand. [21]

Arms:

The arm model provides the orientation, length, width and end points of an arm. When the elbow is not visible, the Leap Motion Controller estimates its position based on past observations as well as the typical human ratio.

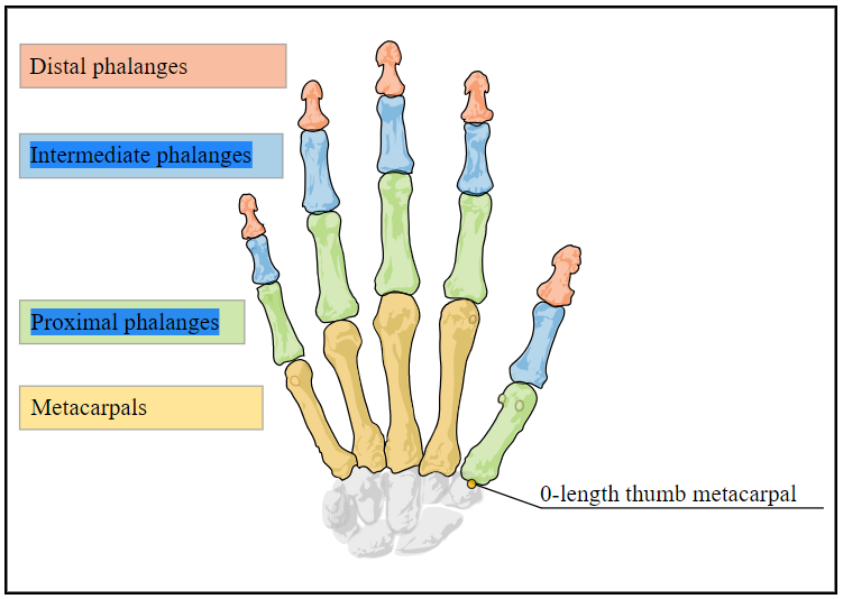
Fingers:

The Leap Motion Controller provides information about each finger on your hand. Fig 3.14. If a finger is not visible, the characteristics are estimated from recent observations and the atomic model of the hand. Fingers are identified by the name of the type, i.e. thumb, index, middle, ring and pinky.

Figure 3.14: Finger tip position and direction vectors provide the position of a finger tip and the general direction in which a finger is pointing [21]

A Finger object provides a Bone object that describes the position and orientation of each atomic bone of the finger. All fingers contain four bones arranged from base to tip.
The bones are identified as [21]:

- Metacarpal – the bone inside the hand connecting the finger to the wrist (except the thumb)
- Proximal Phalanx – the bone at the base of the finger, connected to the palm
- Intermediate Phalanx – the middle bone of the finger, between the tip and the base
- Distal Phalanx – the terminal bone at the end of the finger



This model, anatomically, is not correct in the case of the thumb. A real thumb has one bone less than the other fingers. However, for ease of programming, the Leap Motion thumb model includes a zero length metacarpal bone so that the thumb has the same number of bones in the same index as the other fingers.

3.1.4 Sensor Images

Along with the computed tracking data, you can get the raw sensor images from the Leap Motion cameras.
The image data contains the measured IR brightness values and the calibration data required to correct for the complex lens distortion. You can use the sensor images for augmented reality applications, especially when the Leap Motion hardware is mounted to a VR headset.

3.2 Microsoft Kinect V2

3.2.1 Introduction

The Kinect is a motion sense device developed by Microsoft. It recognizes and tracks the main parts of your body, like arms or legs, as well as face gestures or voice. It's second version its design to be compatible with Windows 10 and has better performance than the previous version. Primarily, it's used for playing video-games, but it's Software Development Kit allows programmers to realize all kinds of applications with it.
A color camera, a depth sensor, which consisting of one Infra-Red projector and one Infra-Red camera, and a microphone array of 4 microphones are build into the Kinect front panel. It has a range of approximately 8 meters, which is a good distance for tracking large movements that involves using the whole body, but from 4 to 8 meters the system starts to receive noise.

3.2.2 Hardware

The color camera, also called RGB sensor, displays a 2D view in three colors: Red, Green and Blue. It has a frame rate of 30 frames per second with a resolution of 640x480 pixels or a frame rate of 12 frames per second with a resolution of 1280x960 pixels.

The depth sensors Infra-Red camera measures the depth of the view by studying the lightened scene that the Infra-Red projector has illuminated. It also has a frame rate of 30 frames per second, but with a resolution of 640x480 or 320x240 pixels.

The sensor can measure distances from 0.5 to 4.5 meters in the default mode, as you can see in Fig 3.15, but can go until 8 meters. The recommended distance, the *sweet* spot, is from 1 to 4 meters. The angle of vision of the Kinect, as shown in Fig 3.16, is of 70 degrees horizontally and 60 degrees vertically.

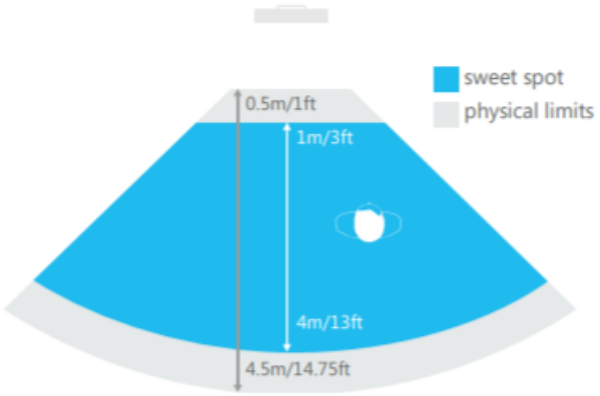


Figure 3.15: Default Mode Range [22]

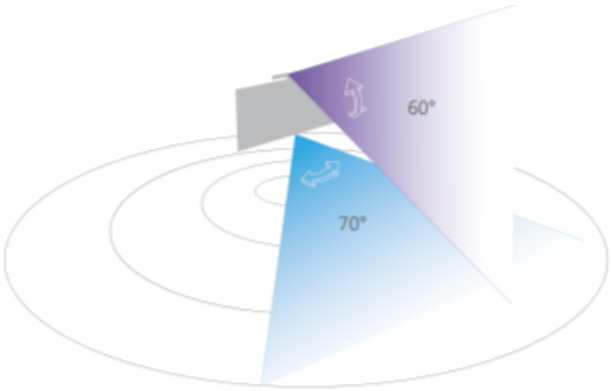


Figure 3.16: Angle of Vision [22]

The array of microphones are used for detecting and recognize voice sound or commands. one of the microphones is located left of the Infra-Red projector while the other three are place on even spaces between them next to the Infra-Red camera. The range of the audio input tat the Kinect can detect goes from -50 to 50 degrees

in front of it, within a 180 degree range with an increment of 5 degrees as shown in Fig 3.17.

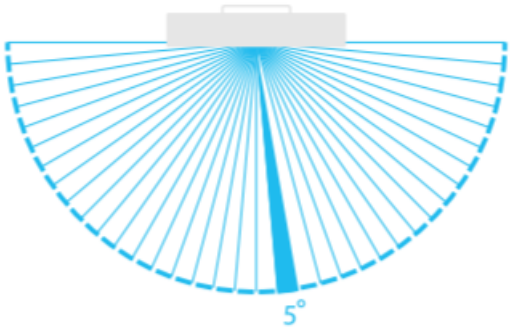


Figure 3.17: Microphone Array Range [22]

3.2.3 Software

Microsoft provides a Software Development Kit (SDK) for programmers to develop new applications with the Kinect V2. The SDK can be implemented in programming languages like C#, C++ or Visual Basic .NET. Also, there are a variety of libraries on the internet which helps the programmer expanding his opportunities for implementing new features that the SDK lacks.

3.2.4 Body Tracking

The Kinect V2 can track up to six people, displaying their skeleton. The skeleton tracking of the Kinect, detects up to 25 joints, if the person is standing, or 10 joints if the person is seated, as you can see on Fig 3.18. The body joints consist of right hand center, right hand thumb, right hand tip, right wrist, right elbow, right shoulder, head, centre of shoulders, neck, left shoulder, left elbow, left wrist, left center hand, left hand thumb, left hand tip, right foot, right ankle, right knee, right hip, spine, centre of hips, left hip, left knee, left ankle and left foot.

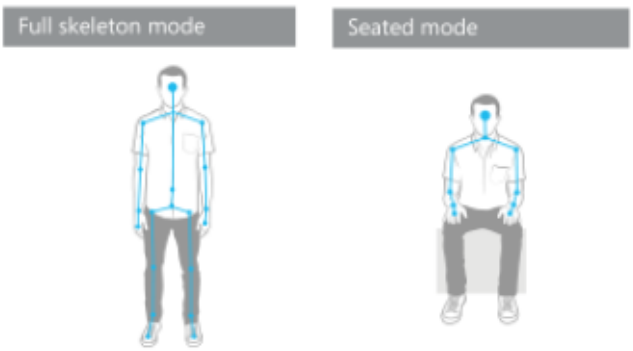


Figure 3.18: Tracked skeleton in two different positions [22]

3.3 Marker Motion Capture

Marker Motion Capture also called Marker Motion Performance or Marker Mocap is a way of recording the movement of an object or a person on a computer using a series of sensors. This technique has been used in a variety of fields such as cinema, video games, the army, for medical purposes...

3.3.1 Markers

The Markers are the sensors used for motion capture. They are small balls that are positioned near the interpreter's joints to identify movement by the positions or angles between them as we can see in the Figure 3.19. These Markers are reflective to be tracked, optimally, by a computer to be used later in the animation. These sensors return their position in space through a vector (X, Y, Z).

Figure 3.19: Different size markers

3.3.2 Recording

Most modern systems can extract the interpreter's silhouette and then calculate all the angles of the joints using a mathematical model in the silhouette. For those movements that cannot be captured by the silhouette, there are hybrid systems that can do both things (Markers and silhouette), having the advantage of using fewer Markers.

3.3.3 Animation

These recordings can be exported as c3d files, which can be loaded into the Blender program. Blender is a software program specially dedicated to modeling, lighting, rendering, animation and creation of three-dimensional graphics [23]. In addition, an avatar can be animated from these files, making the avatar's articulations coincide with those previously recorded by the interpreter. Figure 3.20.

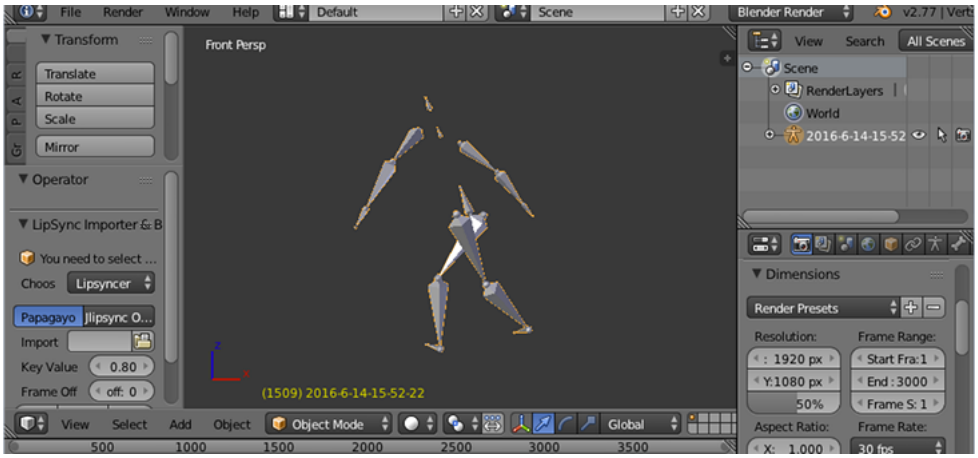


Figure 3.20: Animation in blender from a .c3d file

3.4 TensorFlow and Keras

TensorFlow is an end-to-end open source source library for numerical computation and large-scale machine learning created by the Google Brain Team. TensorFlow can train and run neural networks and it supports production prediction at scale, with the same models used for training. Although TensorFlow is programmed usign Python the mathematical operations are not performed in that langugage. The libraries of transformations that are available through TensorFlow are written as high-performance C++ binaries [24].

Keras is a high-level neural networks API capable of running with TensorFlow as back-end. Keras offers you several modules (as classes or functions) for managing neural networks. It uses *HDF5* files for saving the data of the network and all the acquired knowledge.

3.5 Methodology

For the recognition of sign language, several methods were studied in order to find the best technology capable of recognizing hands for communication with signs, reaching the conclusion that the Leap Motion Controller device is better than other technologies such as the Marker Motion Capture or Microsoft Kinect for static signs, since it can track finger positioning, and the Microsoft Kinect for dynamic gestures, since it has a longer range and can track the hold body of the user.
The project was carried out from a development tool of the company Leap Motion [25] and Microsoft Kinect [22], that allows the creation of applications for the use of the devices. These tools contains documentation, in different languages, to facilitate their use [26].
The static gestures have been captured from the Leap Motion Controller sensor. It tracks hand and finger movements in 3D digital format. It gives a mapping of the gesture in terms of feature points. The positions of the fingertips and palm tips are collected to calculate the euclidean distances between them and then compare with gestures previously recorded in a database from the similarity of the cosine.

Dynamic signs have been captured the same way as the static ones but using the Microsoft Kinect. The Kinect can track the whole upper body of the user and maps the body parts as joints. Then, the euclidean distance is calculated using the coordinates of these joints, comparing them with the signs stored in the database using the Hidden Markov Model.

Regarding to the machine learning part of this project, the very first approach was creating a neural network. After making some research about classification problems, we saw that few people used Support Vector Machine to solve simple classification problems, so we got into it. After some further investigations, we decided not to use Support Vector Machine, because it only have high accuracy if the dataset is simple, and you have a few number of classes or labels. The accuracy on bigger data sets and multilabel classification problems was close to 45%, specifically in sign language classification [27], so we decided to start again with out first approach, neural networks.

3.5.1 Choice of technology

In the first place the Motion Capture technology (with Markers) was chosen as the best for the recording and recognition of gestures because the recordings were very precise thanks to its technology of identification of Markers in addition the files obtained (.c3d) were very versatile to use thanks to that each line contains all the positions of the Markers recorded in a concrete frame, these Markers were ordered along the axis X in the first frame. Because some gestures are dynamic, i.e. a movement is needed to express the gesture, this technology was ideal as the recording field was extensive. But after some tests this method was discarded because to obtain functional recordings had to be recorded many times to get an optimal one. If the markers were not captured by the cameras in some frame these obtained a position of (0, 0, 0) and, therefore, they have made very difficult the obtaining of good recordings. As some of the gestures superimpose fingers or hands, the Markers changed the positions between them being very difficult the cleaning of these. For these reasons Marker Motion Capture was discarded as a useful method for the project.

The Microsoft Kinect technology was also considered because, like the Marker Motion Capture, it could make recordings that needed a large space. In addition the Kinect software provides facial recognition, which is very important when making certain gestures, but due to the lack of precision in the hands and fingers, which was our most important point, and the lack of fluidity offered (compared to other technologies such as Leap Motion) it was discarded as the main technology for static signs but considered for dynamic ones. It offers the possibility of using C# or C++ as the programming language of the application, as well as a variety of environments to implement it.

Finally, the Leap Motion Controller technology was considered for the static signs because the capture of the hands had a very low latency, being considered almost null, and it's precision. This technology differentiates each of the bones of each

finger (as explained above) being able to return any of the positions of these bones with great precision, making it a very versatile technology when performing the recognition. In addition to being a modern technology is low cost, compared to others, which could be, in the future, a way of learning cheap, transportable and easy. It provides software for the development of applications, being able to use a great variety of computer languages such as Java Script, C#, C++, Python, etc, apart from being able to use different environments such as Unity or Unreal Engine among others. While it does not support facial recognition and its field of action is not too wide, this technology was considered to develop the static signs because of its accuracy and low latency which were the most key points.

3.5.2 Static Gestures Recognition

This section have been performed by *Jorge Casas*.

Data Acquisition

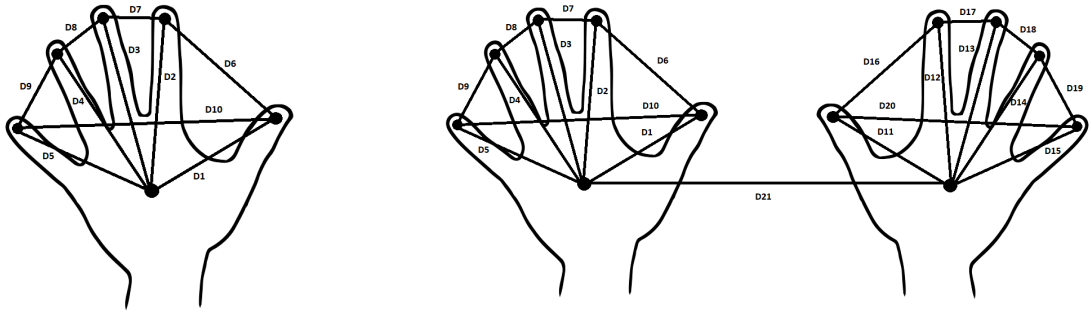
The Leap Motion yields a series of frame by tracking motion of hands and fingers within its view field. The frame data represents a set of tracking data for hands and fingers detected in a single frame. The acquired data through the Leap sensor consists of array of objects to store the physical characteristics of a detected finger such as fingertips. The gestures captured by the Leap controller yields a few key-points corresponding to each hand gesture.[28]

Data extraction

From the sensors of the Leap Motion Controller a set of points and hand vectors relevant for the recognition of gestures are obtained. These points are the end of the distal phalanx, i.e. the tip of the index, middle, ring and pinky fingers and the position of the center of the palm. Figure 3.21 In addition the normal vector to the palm will also be obtained. At first this vector was not used but after several recordings it was verified that the orientation of the hand was necessary to be able to distinguish some gestures from others. These points are returned by the Leap Motion software as vectors (X, Y, Z), which facilitate the use of the data for calculations. In each frame an array of length 18 (6x3) is obtained which contains the positions (X, Y, Z) of all the fingers of the hand and the palm. In case of detecting two hands the vector would be length 36 (12x3). One of the problems that arose during the development of the program was the order of the hands when two hands were detected, since the first 18 values would be equivalent to the first detected. This was modified so that the first 18 values, as long as two hands were detected, were those of the left hand and the next 18 were those of the right hand.

Gesture Recognition

As explained previously, the position (X, Y, Z) of the fingertips, palm centre and palm normal vector were obtained and the euclidean distances between palm centre with each of the five fingers, the distances between adjacent fingers and the distance between thumb and pinky were calculated as it can be seen in Figure 3.21. When two hands were detected the distances of both hands were measured and an additional



(a) One hand distances (b) Two hands distances

Figure 3.21: Euclidean distances calculated for one and two hands recognition

distance was calculated which is the distance between palms. This distance was added later during a testing process in which some very similar gestures were only differentiated by the distance between the hands.

This distance model was obtained from an investigation of hand recognition using the Leap Motion Controller [28], which consisted in the distances between the palm and the five fingers and the distances between adjacent fingers (without including the thumb). The distances between thumb and index finger and thumb and pinky were added to increase the precision of the gestures since some gestures could only be differentiated by the position of the thumb as shown in Figure 3.21. Finally, the orientation of both hands was added, as the Y value of the normal palm vector, to the set of values obtained in each frame in order to further increase the precision and recognition of the different gestures, leaving a vector of 12 values (the distances mentioned above plus the orientation of the two hands) in the case of one hand and a vector of 25 values in the case of two hands (12 values of each hand plus the distance between hands).

These values were compared through algorithm of the cosine similarity, with a database, previously recorded, to obtain how similar was the gesture made with the gestures of the database. A detected gesture with more than 0.93 of similarity with any sign in the data base was considered a well performed gesture. As previously explained these values did not need to be normalized since the similarity of the cosine compares the vectors regardless of their size. The cosine similarity algorithm was used due to the fact that in other sign language recognition investigations it was the most accurate algorithm [28].

This recognition was performed without using Machine Learning or neural networks, but it was used to obtain data for the realization of the neural network. The gestures recognized by the program were stored in a database for future use in Machine Learning.

3.5.3 Dynamic Gesture Recognition

This section have been performed by *Jorge Cristobal*.

Data Acquisition

The kinect can track the body of a user each frame, updating it's coordinates. The data that the Kinect recognize is stored as body joints, e.g. right shoulder or middle spine joints. Like it was said before, the Kinect cameras are located in different positions of the device. This leads in a difference between the coordinates of each camera. Kinect allows to re-coordinate a point tracked by one camera but displayed by another one. The camera used to align the body joints perceived by the Kinect body sensor and the real user was the color camera, since we want the user to see himself doing the sign. Also, for a better performance, the points are drawn, including a line connecting them, recreating the skeleton of the person.

Data extraction

For the dynamic gestures, only the upper body is tracked, since the lower one it's not used at all. The coordinates of the joints tracked are re-mapped and stored as points with coordinates X, Y and Z. A total of 14 points are obtained. Then, the euclidean distance between all the points and the middle spine joint is calculated, storing the result in an array of length 13.

Gesture Recognition

This array is stored every frame in a text file, so a single file contains all the frames of a gesture. This makes it easier for comparing between gestures and recognizing the one that it's been performed. For the comparison, at first it was opted to use the Dynamic Time Warping (DTW). The system would read all gestures in the database and perform the DTW of the sequences, calculating the distance between them and choosing the gesture which distance didn't surpass a threshold. Since the database was small, there was no problem at first, but another solution was chosen since the database should have a bigger size. This alternative was using the Hidden Markov Model (HMM) instead of using the DTW. The HMM reads all the gestures in the database and its trained with that data. Gestures are classified by classes and the HMM chooses the class in which the performed gesture belong. There is a kind of neural network called the Long Short Term Memory (LSTM) networks, which is capable of classify sequences as well as the HMM. Since static signs are classified by a neural network, the LSTM network was chosen as the final classifier for dynamic signs.

3.5.4 Neural Network

This section have been performed by *Javier Luengo*.

Static Signs

As it is said before, the machine learning part in this project is going to be performed by a neural network. The very first part of creating a neural network is to choose an accurate dataset for the problem to solve. Once you have the correct dataset, is time to create the architecture of the network. Once is created and checked that the architecture is optimal for the problem, after some trial and error, the training and validation phase starts. The final step is to create the final model and use it over new data for make predictions.

Data pre-processing

As it was said before, the output in the static gesture recognition part is a vector of 13 values (the distances mentioned above plus the orientation of the two hands and the class, the sign in this concrete project). The network is only going to work with one hand gestures, so the vector of 25 values, (both hands) is not going to be used at this moment, it will be used in future work.

The first step of this phase is to receive each individual vector of the data base and split it into two new vectors, one of them will be the 12 values of the distances and orientations of the hand, and the other one will be its corresponding classes or labels. As Keras works with numpy arrays, it is necessary to convert both arrays.

Now that both are separated numpy arrays, each one of them has to be treated in different ways. The array that stores all the input data of the network, all vectors with 12 values, has to be normalized. The array that stores multi-labels has to be converted in a binary label matrix, so that at prediction time, one assigns the class for which the corresponding model gave the greatest confidence. Once the model is fitted and created, after make any prediction, this binary transformation is inverted, so it is easy for the user to see the real label, or in this case, the sign. Both arrays are ready now to be passed to the network.

Network Architecture

The model of the neural network created for solving the classification of sign language gestures is sequential. This means that is a linear stack of layers. This network has 4 different layers Fig. 3.22:

- ***Input layer:*** This is the very first layer of the net. It communicates the network with the external environment. It contains 128 neurons, the function of this layer is to deal with the inputs, in this case, the dimension of the input is a 12 values vector. The activation function in this layer is "$ReLU$", it will be explained later in this report. This layer also has a dropout of 25% in order to prevent overfitting.

- ***Hidden layer #1:*** This is the second layer of the net, and the first hidden layer. The hidden layers are the ones that connect the input and the output layers, so they transform the inputs that the input layer receives into information that the output layer can use. It contains 64 neurons. The activation function is also "$ReLU$" and it has a dropout of 25%.

- ***Hidden layer #2:*** This is the third layer of the net and the last hidden layer. It contains 32 neurons. As the previous layers, its activation function is "$ReLU$" and it has a dropout of 25%.

- ***Output layer:*** This is the last layer of the net. This layer collects all the information of the net through the output of the last hidden layer and uses it for completing the classification. It contains 17 neurons, one for each class to classify. The output of this layer will be the final classification of the data set received by the input layer. This output layer uses as activation function the Softmax function. That is why the array of labels have been converted into a binary label matrix.

As it is seen in Fig 3.22, in the input and both hidden layers, the activation function used is $ReLU$ 2.6 and the output layer uses *Softmax* 2.7. Both functions are explained in Chapter 3: Theory.

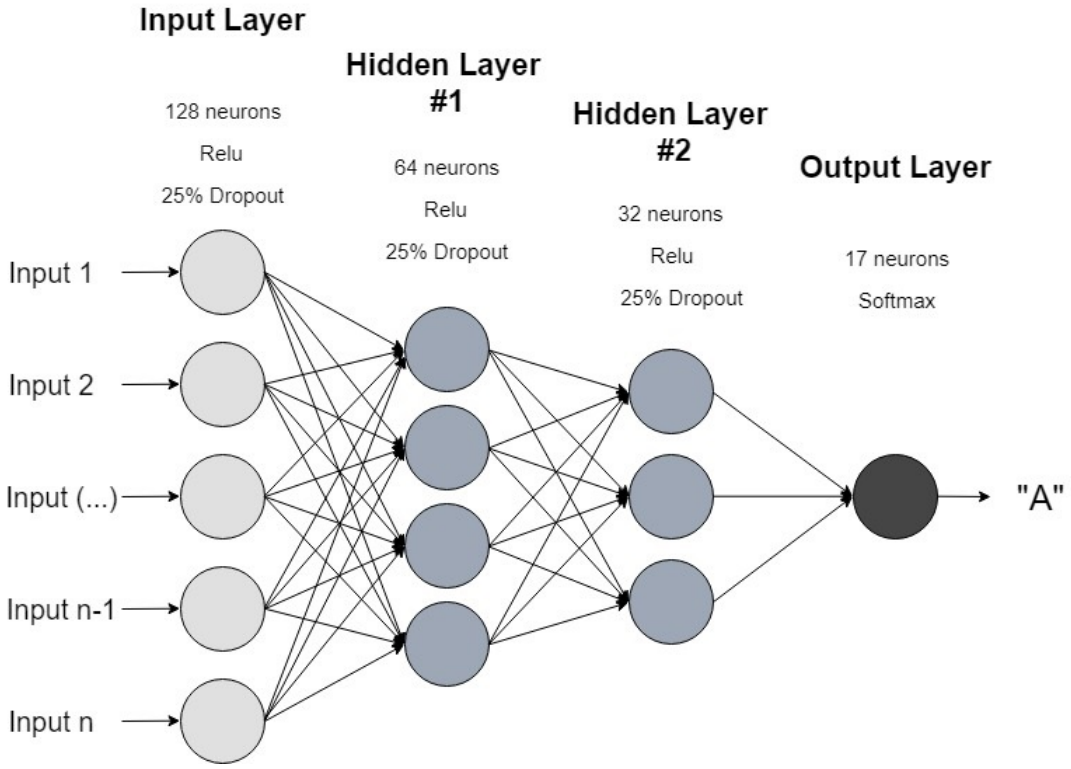


Figure 3.22: Architecture of the neural network

Training, Validation and Testing phase

Once the net is created and the input data set is preprocessed adequately, is time to start training the network. For this phase, is necessary to save the architecture of the network into a model and compile it, so Keras can work with it. At compilation time, two parameters are chosen:

- ***Optimizer:*** The optimizer chosen in this ***Adam***, with a learning rate of *0.0001*. This optimizer is a combination of *RMSprop*, because it uses the squared gradients to scale the learning rate, and *Stochastic Gradient Descent* with momentum because instead using the gradient itself, it uses the moving average of the gradient. It is an adaptive learning rate method, i.e. it computes individual learning rates for different parameters.

- ***Loss Function:*** The loss function chosen is ***Binary Cross-entropy***. This is a modification of *cross-entropy*, but with binary target values Fig. 2.3.

As there is a compiled model, it is possible now to start training it. For this training phase, a data set of 17.000 samples of one-hand gestures. In this training phase it is used a 15% of that data set for make a validation while training, but later on another validation will be performed. At training time, the net uses a batch of 5 samples, this means that the net processes 5 samples independently and in parallel. During the training phase, the net performs the training 10 times over the data set, using different percentage of it for the validation each epoch. The values of the loss and the accuracy are showed at the end of each epoch, so the user can see whether the net is getting better performance or not.

With the model training and validate, now it is possible to save the architecture, the weights and the optimizer state into an *HDF5* file, for its later use. The parameters saved into this file are the parameters of the final model that will be used in the final version of the project. For the testing phase a new data set that the net have never worked on has been used. The testing data set consists in 100 samples of each gesture, a total of 1700 gestures.

Dynamic Signs

Regarding dynamic signs, the procedure of the neural network is the same as for the static one but using the Long Short Term Memory (LSTM) layer. Due to lack of time, only the pre-processing of data has been done, the network architecture is barely finished since the database is small.

Data pre-processing

The input would be the dynamic gesture text file obtained from the Kinect application. This text file would have all frames of the gesture, with an array per frame, composed of 13 euclidean distances. So the input would be a 3D array, as seen in Fig 3.23, containing gestures, frames and distances as the dimensions. Since Keras needs the input to have same length, the number of frames of each gesture is calculated, and then, every array which length is less than that number would be filled with a mask. In this way, all arrays will have the same number of frames.

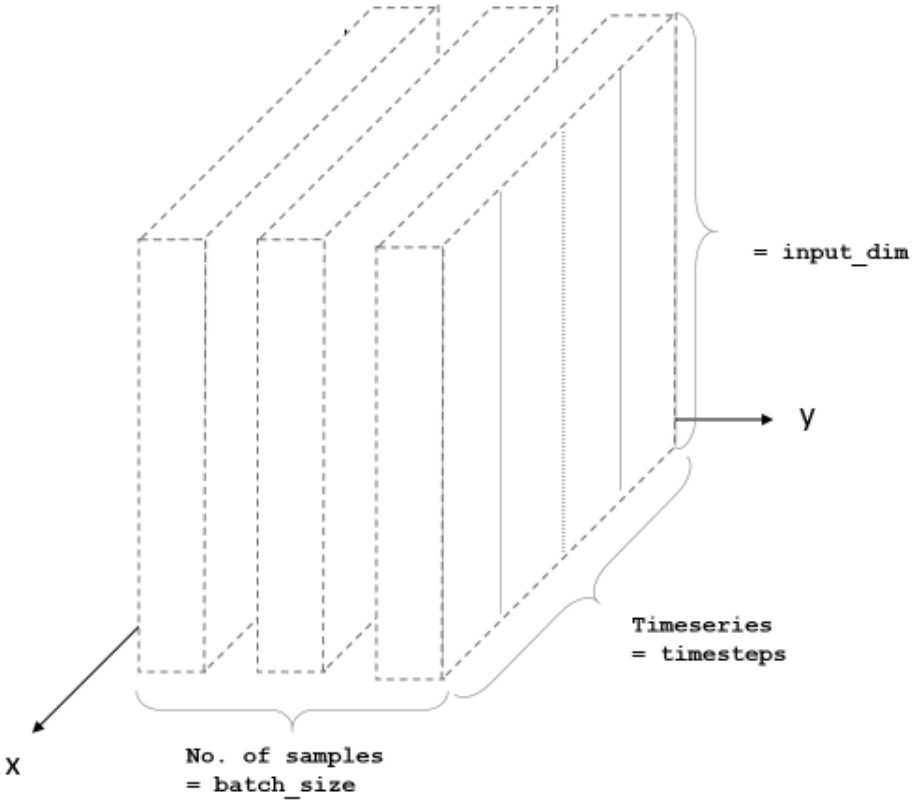


Figure 3.23: Input of a LSTM Network [29]

Network Architecture

For the LSTM to correctly read the gestures, the mask has to be ignored, therefore a Masking layer would be the first one of the model. This layer will take all arrays and read the values, ignoring the mask specified.
Then, as second layer, the LSTM would be present. This layer will take all inputs as sequences and will remember or drop them while it's working, storing information as it is needed. It would be followed by a Output layer, with a neuron for each class.

3.5.5 Final model

The final model is a combination of the static gesture recognition script and the final model of the neural network. The tool used for this combination is a pipeline between both programs. In this pipeline the gesture recognition scripts sends a single sample with the wrong label, inasmuch as the threshold of the recognition script is 0.88 (a low number for accuracy), so that the net predicts this sample and classifies it correctly.

This pipeline works in real time, each sample passed through it belongs to a frame recorded by the Leap Motion. The latency of the final model is almost zero.
For the dynamic gesture recognition there is no implementation for joining both program and neural network because of lack of time and difference between software.

Results

4.1 Scientific Results

4.1.1 Static Gestures

The signs chosen for the recognition of static signs were 17 letters of the international alphabet: [A, B, C, D, E, F, G, H, I, K, L, P, T, U, V, W, Y]. The remaining signs were not chosen because the Leap Motion Controller device could not recognize the signs since some fingers were hidden, such as in the letters M and/or N as shown in Figure 2.1. Others were not selected either because they were dynamic (such as the letters J and Z) or because some signs were too similar for the Leap Motion Controller to differentiate them. 1,000 tests were obtained for each sign to train the neural network, thus producing a total of 17,000 tests. Because a database that satisfies our program was not found, the database had to be created from scratch. Due to the short time for the development of the project these data were obtained from a small number of people recorded the signs mentioned above. To obtain more realistic results a more extensive and varied database is necessary. A total of 10 subjects did tests to check the reliability of the program. These people had to perform the gestures of the international alphabet to test the accuracy of the neural network. These participants had a period to familiarize themselves with the Leap Motion Controller device and were explained how to perform each of the gestures performed in the tests. 100 values of each letter were obtained from these tests.

4.1.2 Dynamic Gestures

The Microsoft Kinect cannot recognize all fingers of the hand and even the one that recognizes (hand tip and thumb) are not accurate. Therefore, the database should only contain large gestures with phrase meaning, such as *How are you?* or *My name is*. This kind of database was not found and, as well as with static gestures, it had to be created. Due to the lack of time and the small number of people, only 5 gestures with 10 samples each were recorded. These gestures are *Are you deaf?, Hello, Please, Nice to meet you,* and *My name is*. It is known that this is a really small database, but for the neural network to work properly, a bigger database of at least 1,000 samples for each gesture is needed, as well as adding up to 10 or 15 gestures.

4.2 Engineering Results

4.2.1 Static Gestures

In the first stage, the main objective of the project was to find the best method for the recognition of gestures, both static and dynamic, of sign language and to be able to represent them using an avatar. As explained in previous chapters, the Leap Motion Controller device has been the best method for recognition due to its low latency (almost zero), its availability of software for the development of applications and its facility to obtain the position of the different bones of the fingers. The objective was to be able to distinguish the 26 letters of the international alphabet and the numbers from 1 to 10. Currently the static gesture recognition software is capable of analyzing and distinguishing 17 letters of the international sign alphabet with an accuracy of almost 100% with the tests and training database mentioned in previous sections.

Two confusion matrix have been created. The first matrix 4.24 shows the accuracy of the net when it reads a data file as a input. In this data file there are 100 samples of each letter. As it can be seen the accuracy is almost 100%, but the net only misclassified the 'U', reading it as 'V'. The second matrix 4.25 shows the accuracy of the net predicting data in real time. The accuracy in this matrix is as good as the other one, but more letters are misclassified. This is because of the Leap Motion Controller precision, because sometimes the gesture recognition is not immediate.

	A	B	C	D	E	F	G	H	I	K	L	P	T	U	V	W	Y
A	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	94	6	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Figure 4.24: Confusion Matrix data file gestures

	A	B	C	D	E	F	G	H	I	K	L	P	T	U	V	W	Y
A	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	99	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	2	0	98	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	3	0	0	97	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	98	0	0	0	2	0	0	0	0	0	0
H	0	0	0	0	0	0	0	97	0	1	0	0	0	2	0	0	0
I	0	0	0	0	5	0	0	0	95	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	99	0	1	0	0	0	0
P	0	3	0	2	0	0	0	0	0	0	0	90	5	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	4	96	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Figure 4.25: Confusion Matrix real time gestures

4.2.2 Dynamic Gestures

The objective was to distinguish between a good range of words and sentences for the recognition of dynamic signs. Currently the system cannot classify the 5 gestures, since the database is really small and the neural network is overfitted. With a larger database, the system could recognize all different sings as long as it does not involve finger recognition.

4.3 Administrative Results

4.3.1 Pre-requirements

Windows

- Python 2.7 (for Leap Motion Controller SDK).
- Python 3.6 or lower (by this date there aren't any later version of Python to run TensorFLow).
- Link for both Python versions.
  `https://www.python.org/downloads/`
- Leap Motion SDK (2.3.1 version). It might not work in later versions.
  Link: `https://developer.leapmotion.com/sdk/v2`
- Tensor Flow 1.6 (2.0 version is not solid).
  Link: `https://www.tensorflow.org/install/source_windows`
- Packages for python:
  - numpy.

  - csv.
  - math.
  - sklearn.
  - keras.

Linux

- Python 2.7 (for Leap Motion Controller. It can be used to run TensorFlow y Keras as well).
- Python 3.6 or lower (for TensorFlow and Keras).
- Link for both Python versions.
  `https://www.python.org/downloads/`
- Leap Motion SDK (2.3.1 version). It might not work in later versions.
  Link: `https://developer.leapmotion.com/sdk/v2`
- Tensor Flow 1.6 (2.0 version is not solid).
  Link: `https://www.tensorflow.org/install/source_windows`
- Packages for python:
  - numpy.
  - csv.
  - math.
  - sklern.
  - keras.

4.3.2 User Guide

Newnet.py

This script has to be used if new sings are added to the network. It creates the neural network used in this project. It is possible to change the parameters of the architecture of the network. If a new network is created is necessary to train again the model. Depending of the changes made in the architecture it would also be necessary to change the parameters of the training phase, as the batch size or the epochs.

Retrain.py

This script is used for re-training the network with new data. The only that have to be modified in this script is the name of the *.txt* in the pre-processing part of the data, that is the database that will be used for the new training phase. If new signs have been added to the net, the have to be also added manually in this script in the label array. In order to retrain the model, the new data base must have the correct shape regarding data format (vectors of 13 values, being the last one the corresponding class, separated by commas), so that the net can work with them.

Possible problems

If the Leap Motion Controller device is not recognized, checking the drivers version could be a solution.
Execute C:/Program Files (x86)/Leap Motion/Core Services/Drivers/dpinst64
This executable can be somewhere else (depends where you install the software).
If the device is still not working check in services that the Leap Motion is running correctly.

Hourly accounts

Hourly accounts are attached at the end of the document as an annex.

Discussion

At the beginning of the project, the goal was to implement a system which could recognize all kinds of signs and an avatar who would teach these signs, but, because of lack of time, only a system who could recognize 17 signs was developed.
One of the main requirement was achieved, a useful method to recognize signs with high precision using accessible technology was found. But other important requirement were not accomplished. The system uses different software for the recognition and classification of the signs. As said before, there is no avatar to reproduce, with an animation, the signs for teaching, as well as not combining the most important aspect of the Kinect with the system, the facial recognition.

Regarding the program itself for static signs recognition, it can be launched in Windows as well as in Linux. But one problem is that in Windows both Python 2.7 and Python 3.6 are needed, because the Leap Motion uses the former and Tensor Flow uses the latter.

The use of Leap Motion helped the develop of this project because of it's variety of programming languages and it's precision. Also, because of it's Software Development ToolKit, which made the creation of applications easier. The choose of tensorflow was also great, since it is a powerful tool for machine learning. For dynamic signs, Kinect was the best option because of it's great range and facial recognition.

All the results were obtained using the database described before and by making tests. These results are outstanding for the static signs recognition, reaching almost a 100% of accuracy, but because of the small database used. For the results of dynamic signs recognition, it is clear that a bigger database is required in order to have any kind of feedback from the neural network. In order to approach to a more realistic outcome, a better and bigger database is needed, as well as making more tests. Another way of improving the system could be by finding a software who could implement both static and dynamic sign recognition together in a single application. When the project was reaching the final stage with the training of the neural network, another approach for solving the problem at stake, the recognition of signs, was taking in mind, but discarded since it involved the start of the project from zero. This approach was the recognition of signs from video sources, using machine learning to train a neural network to classify signs taking video as an input. This could be done by researching on what is called the VVG_16, which is a convolutional network for large-scale image recognition.

Conclusion

Overall, the Leap Motion Controller is a good choice for recognizing signs, as well as using TensorFlow for creating neural networks for training and classifying data. Kinect is not as well as it was thought, despite having a good range and recognizing both body and face, since it can't distinguish between fingers and palm.

Regarding future work, a bigger database is needed in order to get more realistic results. This database should be created with a lot of people so different hand sizes are taking in count. Also, finding a possible way to record dynamic and static signs using just the Leap Motion Controller should be the main priority. Another solution could be forging both Leap Motion and Kinect software to implement the Kinect facial recognition. Due to the results obtained in the tests of the final model, the remaining letters of the international alphabet and numbers could be added to the database.

As it was said before, other possible alternative for the sign recognition is the processing of video recordings, where the VGG-16 Convolutional Network could be used. Other solution is changing the algorithms for the extraction of input data, so, instead of using the euclidean distance, other values could be taken as an input, like the angles of the arms against the body.

Bibliography

[1] EcuRed. *Lenguaje de señas*. 2014. URL: https://www.ecured.cu/Lenguaje_de_se%C3%B1as.

[2] Norges Døveforbund. *Hva er tegnspråk?* URL: https://www.doveforbundet.no/tegnsprak/hva.

[3] Norges Døveforbund. *Håndalfabetet*. URL: https://www.doveforbundet.no/tegnsprak/handalfabetet.

[4] Wolfram MathWorld. *Euclidean Space*. 2019. URL: http://mathworld.wolfram.com/EuclideanSpace.html.

[5] RevolEdu. *Euclidean Distance*. 2017. URL: https://people.revoledu.com/kardi/tutorial/Similarity/EuclideanDistance.html.

[6] Gustavo E. A. P. A. Batista Diego F. Silva. "Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation". In: *ResearchGate* (2013).

[7] H. Sakoe and S Chiba. *Dynamic programming algorithm optimization for spoken word recognition*. URL: https://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm.

[8] MachineLearninPlus. *Cosine Similarity – Understanding the math and how it works*. URL: https://www.machinelearningplus.com/nlp/cosine-similarity/.

[9] Matt Delbridge. *Motion capture in performance*. London: Palgrave Pivot, 2015.

[10] Rohith Gandhi. *Support Vector Machine—Introduction to Machine Learning Algorithms*. 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[11] Thomas Bolander. "Synergies Between Symbolic and Sub-symbolic Artificial Intelligence". In: *Current Trends in AI, 23 November 2016* (2016).

[12] BiologyReference. *Neuron*. URL: http://www.biologyreference.com/Mo-Nu/Neuron.html.

[13] Suzana Herculano-Houzel. "The Human Brain in Numbers: A Linearly Scaled-up Primate Brain". In: *Frontiers in Human Neuroscience* (2009).

[14] José M. Valls Inés M. Galván. "Introducción a las Redes de Neuronas". In: *Universidad Carlos III de Madrid* (2017).

[15] Danqing Liu. *A Practical Guide to ReLU*. 2017. URL: https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7.

[16] Junping Du Binghui Chen Weihong Deng. *Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation*. 2017. URL: https://www.researchgate.net/publication/319121953_Noisy_Softmax_Improving_the_Generalization_Ability_of_DCNN_via_Postponing_the_Early_Softmax_Saturation.

[17] Victor D. Fachinottia Facundo Brea Juan M. Gimeneza. "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". In: *ResearchGate* (2017).

[18] Raúl Gómez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. 2018. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[19] Zoubin Ghahramani. "An Introduction to Hidden Markov Model and Bayesian Networks". In: *ResearchGate* (2001).

[20] Terencehonles. *Hidden Markov Model*. 2009. URL: https://en.wikipedia.org/wiki/Hidden_Markov_model.

[21] Leap Motion SDK. *API Overview*. 2017. URL: https://developer-archive.leapmotion.com/documentation/python/devguide/Leap_Overview.html.

[22] Microsoft Corporation. *Human Interface Guidelines, version 2.0*. 2014. URL: http://download.microsoft.com/download/6/7/6/676611b4-1982-47a4-a42e-4cf84e1095a8/kinecthig.2.0.pdf.

[23] Ton Roosendaal. *Blender*. 2018. URL: https://www.blender.org/.

[24] Serdar Yegulalp. *TensorFlow, the machine learning library*. 2018. URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

[25] Leap Motion. *Leap Motion / Developer*. 2019. URL: https://developer.leapmotion.com/.

[26] Leap Motion. *Leap Motion / Developer*. 2019. URL: https://developer-archive.leapmotion.com/documentation/python/index.html.

[27] Jamia Millia Islamia Mr Sarfaraz Masood. "Sign Language Character Recognition". In: *ResearchGate* (2016).

[28] Archana Ghotkar Chetna Naidu. "Hand Gesture Recognition Using Leap Motion Controller". In: *International Journal of Science and Research (IJSR)* (2016).

[29] Shiva Verma. *Understanding Input and Output shapes in LSTM*. 2019. URL: https://medium.com/@shivajbd/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e.

Annexes

		Hourly accounts *Jorge Casas Guerrero*
Date	Hours	Description
01/02/2019	4	Investigation about possible methods
02/02/2019	4	Investigation about possible methods
03/02/2019	6	Investigation about possible methods
04/02/2019	6	Investigation about possible methods
05/02/2019	6	Investigation about possible methods
06/02/2019	6	Investigation about possible methods
07/02/2019	1,5	Meeting with the teachers
07/02/2019	4	Investigation and animation in blender
08/02/2019	4	Animation in blender with c3d files
12/02/2019	4	Animation in blender with c3d files
13/02/2019	4	Meeting with the teachers and animation with new c3d files
14/02/2019	4	Blender animation and openpose
27/02/2019	4	Openpose and Leap Motion Controller
28/02/2019	4	Leap motion controller
01/03/2019	5	Leap motion controller
04/03/2019	5	Leap motion controller
05/03/2019	5	Leap motion controller
06/02/2019	5	Leap motion controller
08/02/2019	5	Leap motion controller
10/02/2019	5	Leap motion controller
11/03/2019	5	Leap motion controller
12/03/2019	5	Leap motion controller
13/03/2019	2	Meeting with the teachers
14/03/2019	4	Leap motion controller
15/03/2019	4	Leap motion controller
19/03/2019	4	Leap motion controller
20/03/2019	4	Leap motion controller
25/03/2019	3	Question research presentation
26/03/2019	4	Leap motion controller
26/03/2019	1	Question research presentation
27/03/2019	5	Leap motion controller
01/04/2019	6	Leap motion controller
02/04/2019	4	Leap motion controller and dynamic gestures investigation

Date	Hours	Description
03/04/2019	4	Dynamic gestures investigation
04/04/2019	1	Presentation and meeting progress
04/04/2019	4	Leap motion controller tensorflow
05/04/2019	5	Investigation machine learning
08/04/2019	4	Investigation machine learning
09/04/2019	4	Tensorflow
10/04/2019	6	Tensorflow
11/04/2019	5	Tensorflow
12/04/2019	4	Meeting and working on future goals
16/04/2019	4	Leap motion controller (problems with gestures)
22/04/2019	4	Leap motion controller
22/04/2019	2	Report
23/04/2019	4	Leap motion controller
23/04/2019	1	Report
25/04/2019	4	Leap motion controller
25/04/2019	4	Report
29/04/2019	5	Report
30/04/2019	1	Report
30/04/2019	4	Leap motion controller
01/05/2049	5	Leap motion controller
02/05/2019	5	Leap motion controller
04/05/2019	5	Leap motion controller
04/05/2019	2	Report
06/05/2019	8	Leap Motion Controller
07/05/2019	1	Report
07/05/2019	6	Leap Motion Controller
08/05/2019	4	Report
08/05/2019	4	Leap motion controller
09/05/2019	6	Leap motion controller
10/05/2019	5	Leap motion controler (pipeline)
10/05/2019	3	Report
14/05/2019	6	Report
14/05/2019	2	Leap motion controller
15/05/2019	3	Leap motion controller and tensorflow
15/05/2019	4	Report
21/05/2019	3	Geting results
21/05/2019	2	Report
22/05/2019	3	Report
23/05/2019	3	Results
24/05/2019	3	Results
26/05/2019	3	Results
27/05/2019	3	Results
28/05/2019	3	Results
29/05/2019	3	Results
30/05/2019	3	Results
31/05/2019	3	Presentation

		Hourly accounts *Javier Luengo Peñas*
Date	Hours	Description
04/02/2019	6	Investigation about possible methods
05/02/2019	6	Investigation about possible methods
06/02/2019	6	Investigation about possible methods
07/02/2019	1,5	Meeting with the teachers
08/02/2019	5	Investigation about possible methods
12/02/2019	4	Investigation about possible methods
13/02/2019	2	Meeting with the teachers
14/02/2019	4	Investigation about possible methods
17/02/2019	4	Investigation about possible methods
18/02/2019	4	Investigation about possible methods
20/02/2019	4	Investigation about possible methods
21/02/2019	4	Investigation about possible methods
24/02/2019	3	SVM investigation
25/02/2019	3	SVM investigation
26/02/2019	3	SVM investigation
27/02/2019	3	SVM investigation
04/03/2019	4	Neural Networks investigation
05/03/2019	4	Neural Networks investigation
06/03/2019	5	Neural Networks investigation
08/03/2019	4	Neural Networks investigation
10/03/2019	5	Neural Networks investigation
11/03/2019	5	Neural Networks investigation
12/03/2019	5	Neural Networks investigation
13/03/2019	2	Meeting with the teachers
15/03/2019	5	Neural Networks investigation
18/03/2019	4	Neural Networks investigation
19/03/2019	4	Tensorflow
20/03/2019	4	Tensorflow
21/03/2019	3	Tensorflow
25/03/2019	3	Question research presentation
26/03/2019	4	Tensorflow
26/03/2019	1	Question research presentation
27/03/2019	5	Tensorflow
01/04/2019	5	Tensorflow
02/04/2019	4	Tensorflow
03/04/2019	4	Dynamic gestures investigation
04/04/2019	1	Presentation and meeting progress
04/04/2019	4	Dynamic gestures investigation
05/04/2019	4	Tensorflow
08/04/2019	4	Tensorflow
09/04/2019	4	Tensorflow
10/04/2019	4	Tensorflow
11/04/2019	5	Tensorflow
12/04/2019	4	Meeting and working on future goals
16/04/2019	4	Tensorflow

Date	Hours	Description
22/04/2019	4	Tensorflow
22/04/2019	2	Tensorflow
23/04/2019	3	Tensorflow
23/04/2019	2	Tensorflow
25/04/2019	3	Tensorflow
25/04/2019	4	Report
29/04/2019	5	Report
30/04/2019	2	Tensorflow
30/04/2019	4	Tensorflow
01/05/2049	4	Leap motion controller
02/05/2019	5	Leap motion controller
04/05/2019	2	Report
06/05/2019	4	Tensorflow
07/05/2019	2	Tensorflow
07/05/2019	4	Report
08/05/2019	3	Report
08/05/2019	4	Tensorflow
09/05/2019	5	Tensorflow
10/05/2019	5	Tensorflow
14/05/2019	4	Report
14/05/2019	2	Report
15/05/2019	3	Leap motion controller and tensorflow
15/05/2019	4	Report
21/05/2019	3	Geting results
21/05/2019	2	Report
22/05/2019	3	Report
23/05/2019	3	Results
24/05/2019	2	Report
24/05/2019	2	Results
24/05/2019	3	Report
26/05/2019	3	Results
28/05/2019	2	Report
29/05/2019	3	Results
30/05/2019	3	Results
31/05/2019	3	Presentation
01/06/2019	2	Report

		Hourly accounts *Jorge Cristobal Martin*
Date	Hours	Description
01/02/2019	4	Investigation about possible methods
03/02/2019	6	Investigation about possible methods
04/02/2019	5	Investigation about possible methods
06/02/2019	4	Main Report Translation
07/02/2019	1,5	Meeting with the teachers
07/02/2019	4	Investigation about cameras
10/02/2019	4	Investigate about Kinect V2
11/02/2019	2	Investigate about Kinect V2
13/02/2019	4	Meeting with the teachers and Learning Visual Studio
14/02/2019	4	Learning Visual Studio with Kinect
27/02/2019	4	Learning Visual Studio with Kinect
28/02/2019	6	Kinect V2
02/03/2019	4	Kinect V2
04/03/2019	5	Kinect V2
05/03/2019	3	Kinect V2
07/02/2019	5	Kinect V2
08/02/2019	4	Kinect V2
10/02/2019	6	Kinect V2
11/03/2019	5	Kinect V2
12/03/2019	4	Kinect V2
13/03/2019	2	Meeting with the teachers
15/03/2019	5	Kinect V2
17/03/2019	4	Kinect V2
19/03/2019	5	Kinect V2
20/03/2019	4	Investigation about Face Recognition with Kinect
22/03/2019	3	Question research presentation
24/03/2019	4	Kinect V2
26/03/2019	1	Question research presentation
27/03/2019	6	Kinect V2
01/04/2019	6	Kinect V2
02/04/2019	4	Investigation about Face Recognition with Kinect
03/04/2019	4	Face Recognition with Kinect V2
04/04/2019	1	Presentation and meeting progress
05/04/2019	4	Kinect V2
06/04/2019	5	Investigating machine learning
08/04/2019	4	Investigating machine learning

Date	Hours	Description
09/04/2019	4	Dynamic Time Warping
10/04/2019	6	Dynamic Time Warping
11/04/2019	5	Hidden Markov Model
12/04/2019	4	Meeting and working on future goals
19/04/2019	5	Hidden Markov Model
20/04/2019	4	Hidden Markov Model
21/04/2019	2	Report
22/04/2019	4	Hidden Markov Model
23/04/2019	1	Report
24/04/2019	4	Kinect V2
25/04/2019	4	Report
28/04/2019	5	Report
30/04/2019	1	Report
30/04/2019	4	Kinect V2
01/05/2049	5	Kinect V2
02/05/2019	3	Kinect V2
03/05/2019	5	Kinect V2
05/05/2019	2	Report
06/05/2019	7	Kinect V2
07/05/2019	2	Report
07/05/2019	5	Kinect V2
08/05/2019	4	Report
08/05/2019	4	LSTM Neural Network
09/05/2019	6	LSTM Neural Network
11/05/2019	5	LSTM Neural Network
12/05/2019	3	Report
14/05/2019	6	Report
15/05/2019	4	Kinect V2
17/05/2019	3	Kinect V2
18/05/2019	4	Report
21/05/2019	2	Report
22/05/2019	3	Database
24/05/2019	3	Database
26/05/2019	4	Report
27/05/2019	3	Report
29/05/2019	3	Report
31/05/2019	2	Presentation