UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. De Telecom., Sonido e Imagen

"Gesture recognition using a depth sensor and machine learning techniques"

FINAL BACHELOR THESIS

Author:
MARINA BALLESTER RIPOLL

Supervisors:
Prof. Dr. Eng./ University of Tsukuba Jens Herder
Prof. Dr.–Ing. Thomas Bonse

DÜSSELDORF, 2016

To my parents, who have trusted me
in any decision I've ever made.
And to my brother,
who has always been a role model to me.

STATEMENT OF ORIGINALITY

I declare that I am the sole author of this bachelor thesis titled *Gesture recognition using a depth sensor and machine learning techniques* .

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institution.

The data, concepts, softwares and tools that have been taken directly or indirectly from other sources have been acknowledged and referenced.

DATE

SIGNED

Abstract

Advances in depth sensing provide great opportunities for the development of new methods for human computer interactivity. With the launch of the Microsoft Kinect sensor, high-resolution depth and visual sensing has become available for widespread use. As it is suitable for measuring distances to objects at high frame rate, such kind of sensors are increasingly used for 3D acquisitions, and more generally for applications in robotics or computer vision. The aim of this survey is to implement a gesture recognition system using the Kinect version 2 of Microsoft in order to interact with a virtual TV weather studio. The Kinect sensor was used to build up a dataset, which contains motion recordings of 8 different gestures and was build up by two different gesture training machine learning algorithms. Then, the system is evaluated in a user study, which allows a direct comparison and reveals benefits and limits of using such technique. Finally, it is given an overview of the challenges in this field and future work trends.

Keywords: Gesture recognition, Interaction, Kinect, Machine learning, Virtual studio

Los avances en los sensores de profundidad ofrecen grandes oportunidades para el desarrollo de nuevos métodos para la interactividad computadora-humano. Con el lanzamiento del sensor Kinect de Microsoft, la detección de profundidad de alta resolución se ha convertido en un componente disponible para el uso generalizado. Como es adecuado para medir distancias a objetos a alta velocidad, este tipo de sensores se utilizan cada vez más para adquirir información 3D, y más en general para aplicaciones en robótica o en visión artificial. El objetivo de este estudio es implementar un sistema de reconocimiento de gestos utilizando la Kinect versión 2 de Microsoft con el fin de interactuar con un estudio virtual de TV. El sensor Kinect se utilizó para construir una base de datos, que contiene grabaciones de movimientos para 8 gestos distintos y fue entrenado por dos algoritmos diferentes de aprendizaje de máquinas. A continuación, el sistema se evaluó con un conjunto de usuarios en un estudio virtual, lo que permite una comparación directa y revela los beneficios y los límites de la utilización de tal técnica. Por último, se da una visión general de los retos en este campo y futuras líneas de trabajo.

Palabras clave: Reconocimiento de gestos, Interacción, Kinect, Aprendizaje de máquinas, Estudio virtual

Els avenços en els sensors de profunditat ofereixen grans oportunitats per al desenvolupament de nous mètodes per a la interactivitat ordinador-humà. Amb el llançament del sensor Kinect de Microsoft, la detecció de profunditat d'alta resolució s'ha convertit en un component disponible per a l'ús generalitzat. Com és adequat per mesurar distàncies a objectes a alta velocitat, aquest tipus de sensors s'utilitzen cada vegada més per adquirir informació 3D, i més en general per a aplicacions en robòtica i en visió artificial. L'objectiu d'aquest estudi és implementar un sistema de reconeixement de gestos utilitzant la Kinect versió 2 de Microsoft per tal d'interactuar amb un estudi virtual de TV. El sensor Kinect es va utilitzar per construir una base de dades, que conté gravacions de moviments per a 8 gestos diferents i va ser entrenat per dos algoritmes diferents d'aprenentatge de màquines. A continuació, el sistema es va avaluar amb un conjunt d'usuaris en un estudi virtual, el que permet una comparació directa i revela els beneficis i els límits de la utilització de tal tècnica. Finalment, es dóna una visió general dels reptes en aquest camp i futures línies de treball.

Paraules clau: Reconeixement de gestos, Interacció, Kinect, Aprenentatge de màquines, Estudi virtual

Contents

List of Figures

Chapter 1

Introduction

Kinect is a RGB sensor providing synchronized colour and depth images. It was initially designed to capture users' body motions for interaction with video games and used as an input device by Microsoft for the Xbox game console. Nevertheless, the computer vision society extended this low-cost depth sensing technology of Kinect v1 far beyond gaming. With a 3D human motion capturing system, it enables interactions between users and other devices or applications without the need to touch a controller so it was quickly repurposed to support a diverse array of new applications ranging from the medical field to robotics.

The next generation, the Kinect v2 sensor, was completely redesigned to use time-of-flight ranging: It bathes the scene with strobed infrared light, and records how long it takes the bursts of light to return to each pixel. It can produce high-resolution (512x424) depth images at 30 frames per second, with roughly an order of magnitude greater depth accuracy than the first version of Kinect, which used structured pattern technology in order to extract the depth information.

With Kinect, people are able to interact with games, devices and other applications simply by performing gestures infront of the sensor in a natural way. The key enabling technology is human bodylanguage understanding; the computer must first understand what a user is doing before it can respond. The Kinect sensor lets the computer directly sense the third dimension (depth) of the users and the environment, making the task much easier. It also knows who they are when they walk up to it, and can interpret their gestures and translate them into a format that developers can use to build new experiences.

In this bachelor thesis will be implemented an application where it will be used a Kinect v2 in order to interact with a virtual weather TV studio and it will be compared with other previous designed gesture control armband system: the Myo bracelet. The objective of this thesis is to evaluate how the Kinect sensor can detect a database of gestures previously recorded by sample users. The Adaptive Boost and the Random Forest Regression algorithms through machine learning were used in order to classify such gestures. Then, it was built a bridge to translate the performed gestures into Open Stage Control (OSC[1]) messages witch were sent to the graphic environment. Those messages initialized the corresponding animations.

This thesis is organized as follows. Firstly, it is discussed the state of the art of the gesture recognition methods where it is talked about several techniques ranging from heuristic to machine learning methods. Secondly, it is discussed the mechanism of the Kinect sensor taking both hardware and software into account. The purpose is to answer what signals the Kinect can output, and what advantages the Kinect offers compared to conventional cameras in the context of several classical vision problems. From section 8 to section 12, it is given technical overviews of the process followed to implement the application to interact with the virtual studio. Later, is compared with the Myo gesture control technology. In section 15 and 16 will be discussed the conclusions of the present work.

[1] `http://opensoundcontrol.org`

1.1. Mixed Reality

1.1.1. Definition

Mixed Reality technology works halfway between real and virtual scenarios providing the suitable tools to link both of them. It presents a case where the real world is attached to a virtual scenery containing virtual objects and are presented in a single display. In this kind of technologies, the user needs the system to respond in real-time in order to interact with virtual scenarios in the same way they interact with real worlds, thus making the interaction task much more natural and reducing training.

The computer-generated virtual objects must be set to the real world in all dimensions. If there are errors in the setting, the user will not have the perception to see both images, virtual and real, merged. In addition, the adjustment of the images must be correct at all times, even when the user is moving, changes in vision due to movement must be taken into account and make the necessary operations to the situation of virtual objects.

A way of interacting between the real world and the virtual one, is by performing natural and intuitive gestures, which should be handy, fast-performing and effective. Is because of this that gesture recognition techniques become a very important issue in this field of the technology. Although, there are two different ways for collecting the required data to perform recognition: the glove-based solution and the vision-based solution. In this survey it will be only talked about this last one.

1.1.2. Vision-based systems

Vision based techniques are non invasive and based on the way human beings perceive information about their surroundings. Although it is difficult to design a vision based interface for generic usage, yet it is feasible to design such an interface for a controlled environment.

Recognizing gestures involve handling a considerable number of degrees of freedom, huge variability of the 2D appearance depending on the camera view point (even for the same gesture), different silhouette scales (i.e. spatial resolution) and many resolutions for the temporal dimension (i.e. variability of the gesture speed). Figure 1.1 shows a flow diagram of a typical gesture recognition strategy.

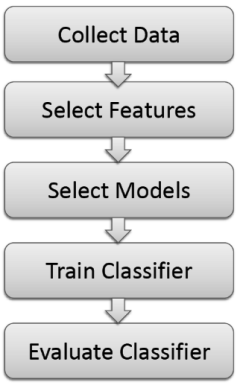

Figure 1.1.: Typical computer vision based gesture recognition approach. [1]

In the data collection step, takes place the training and testing data which must be sufficiently representative. Afterwards, it's made the selection of discriminative descriptors which will have to have similar values for the same class of gestures and different for others as well as invariant to translation, rotation and scale. Also, these descriptors should be robust against occlusion, distortion, noise, deformation and changes in the environment. So it should be chosen a set as uncorrelated as possible. Once the features are selected, it will have to be chosen a decision model: statistical, algorithmic, neural, etc. Afterwards, will take place the training phase of the chain which can be a supervised or an unsupervised kind of training. In the first one, a human person would have to classify the gestures and locate them into a specific class. In the unsupervised training, the decision model conforms groups of gestures autonomously.

The last stage of the gesture recognition proceeding, will take place the evaluation of the descriptors and training that has been made which could result into a possible modification of the model in order to optimize the system.

In order to detect and classify the gestures, several feature extraction techniques have been used to extract the features of the gesture images. These techniques include Orientation Histogram [18], Wavelet Transform [19], Fourier Coefficients of Shape [20], Zernike Moment [21], Gabor filter [22, 23], Edge codes [24], Hu Moment [25], Geometric feature [26] and Finger-Earth Mover's Distance [27]. Most of these features extraction methods have some limitations. In orientation histogram for example, the algorithm employs the histogram of local orientation. This simple method works well if example of the same gesture map to similar orientation histograms, and different gestures map to substantially different histograms. Although this method is simple and offers robustness to scene illumination changes, its problem is that the same gestures might have different orientation histograms and different gestures could have similar orientation histograms which affects its efficiency. Another feature extract method used in vision-based gesture recognition technique is to extract the features from color images where they present a real-time static isolated gesture recognition application using a Hidden Markov Model approach. The features of this application were extracted from gesture silhouettes using color segmentation. The problem of this method is that may be the presence of skin-colored objects in the background that could affect the performance of the system. This, one of the main weaknesses process, if the background has color properties similar to the skin. The feature extraction analysis technique is usually followed by the classification method, which uses the extracted feature vector to classify the gesture image into its respective class. Among the classification methods it can be founded: Nearest neighbour [18], Artificial Neural Network, Support Vector Machines and Hidden Markov Models.

As an example of classification methods, Nearest Neighbour classifier is used as body-part recognition method in combined with Modified Fourier Descriptors to extract features of the a part of the body shape. The system involves two phases: training and testing as in ML. The system stores the carrier coefficients of the body-part shape in the training set, and in the running phase, the computer compares the current hand shape with each of the stored shapes through the coefficients. The best matched gesture is selected by the Nearest Neighbour method using the MED distance metric.

However, vision-based gesture recognition systems that use feature extraction methods suffer from working under different lighting conditions as well as the scaling, translation, and rotation problems. Later in this survey, it will be talked further about this techniques.

Chapter 2

Related research to vision-based gesture recognition

2.1. Single camera

The first instance of a gesture recognition system that totally relied on computer vision was reported by Rhg and Kanade in 1993 [28]. Their system stood out from similar work of gesture recognition research as the user could use this system without the requirement to wear any glove or colored markers as it was used to do before. The system was called DigitEyes [28]. They demonstrated hand tracking at speeds up to 10 Hz using line and point features extracted from grey scale images of ordinary unmarked hands. Most previous real-time visual 3D tracking work had addressed objects with 6 or 7 spatial degrees of freedom by then. Rhg and Kanade presented tracking results for branched kinematic chains with as many as 27 Degrees of Freedom for human hand model which resulted in a highly articulated model. The success of this computer vision approach was mainly due to their ability to extract simple and useful features from human hand, which was in its infancy by then. In order to avoid occlusion across complicated backgrounds, they used two cameras effectively. Also in 1993, Darrel and Pentland developed a method for learning, tracking, and recognizing human gestures using a view-based approach to model articulated objects. In their approach, objects were represented using sets of view models and space-time patterns such as gestures were matched to stored gesture patterns using dynamic time warping.

Because the parameter values underlying the pose are associated with the set of correlation scores, a gesture can be characterized by the pattern of view model scores directly. Recognizing a gesture requires determining which stored gesture most closely matches the observed space-time pattern of model scores. In other words, rather than transforming sets of view model scores to parameters and then matching in parameter space, simple matching in the model-score space directly is sufficient. Furthermore, both view models and view predictions are learned from examples.

The research carried out around the early 1990s always suffered from low camera resolution to inadequate computing power. The features that would be derived from a single camera not always resulted in successful outcomes for many research projects. Utsumi et al. team countered this trend using a stereo rig in order to determine the Centre Of Gravity of the palm of the hand and to calculate the 3D positions of finger tips. However, they failed to implement the system in real-time, yet they were successful in estimating the center of gravity and fingertips location which were later used by other researchers as stepping stone to research on gesture recognition.

Freeman et al. in 1995 developed a new set of stable features using orientation histograms [2]. These offered a very stable and unique set of features which were very robust for hand gesture recognition. Figure 2.1 shows that gestures look different to the naked eye under different color schemes, yet this histogram of oriented gradients are able to provide features that are invariable to different levels of illumination. Their research heralded a new era where stable features were the dominant factor in designing robust gesture recognition. The research conducted in the following years was keen to find unique but stable features so that the recognition performance would increase.

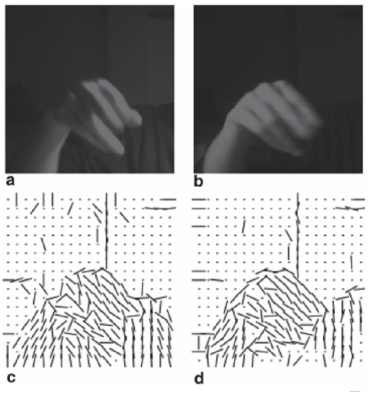

Figure 2.1.: Showing the robustness of local orientation to lighting changes. Pixel intensities are sensitive to lighting changes. a and b are same gestures under two different lighting conditions. (c) and (d) are the orientation maps of these (a) and (b) which are much more stable. [2]

As showed in Figure 2.1, in gestures with movement it's needed a motion detection system to extract a moving image sequence and motion-based visual attention which identifies windows that include moving object. They mapped this temporal window to a standard temporal length to form "motion clip". The speed information was available from the mapping performed in this stage. The second step was object recognition followed by segmentation. During segmentation, the object of interest was framed using a rectangular window from each image in the sequence and then was mapped to a fixed size. This means that the image was normalized in order to make the object of interest appear at a standard position with a standard size. Afterwards they simplified the processing by assuming a simple background which eliminated the need to deal with skin tone regions in their work. They used Most Expressive Features and Most Discriminating Features approach to find methods for good classification results. The reported best performance was 96% of confidence.

Some of the many drawbacks using a single camera include occlusion and the difficulty in removing the cluttered background. Many researchers around 1998 started using multiple cameras to mitigate the effect of occlusion which resulted in many incorrect classifications. Such erroneous classifications often led to negative reviews by the research community as poor results offered no solution for the progress of the research. Even though, the multiple cameras offered better results, having multiple cameras which were physically separated, created cluttered solution and solved the problem of occlusion when a gesture can be seen from many angles which are not possible to be detected by a single camera. However, as obvious, multiple cameras need calibration and other constraints to determine that multiple view belong to the same gesture at a given time. When two cameras are used, stereo vision strategies easily establish the correspondence problem.

2.2. Multiple camera

In 1998, Segen and Kumar of Bell Laboratories developed a vision based 3D hand interface for human machine interaction [3]. Their setup consisted of stereo system to extract the x, y, z location of pointing fingers so that its location could be used to interact with a computer interface displayed in front of the user. The system used 60 frames per second and was very responsive. The method responded to 4 types of gestures: Point, Reach, Click and Ground as illustrated in Figure 2.2 . The system allowed them to use different combinations of those gestures to manipulate Virtual Reality-type games. Segen and Kumar demonstrated that multiple applications can be run with few basic gestures with an appropriate

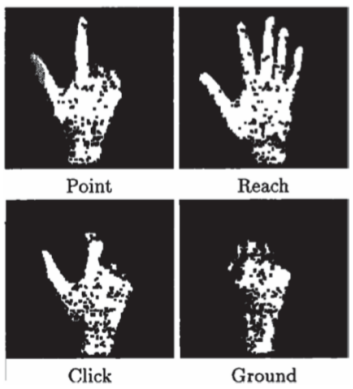

Figure 2.2.: Gestures used in virtual reality by Segen and Kumar. [3]

computer interface. This leaded to the development of many gestural interfaces so far. Similar to the work carried out by Segen and Kumar, Utsumi and Ohya proposed a gesture tracking system using multiple cameras in 1999 [29]. The system that they proposed tracked 3D position of the posture of the hand with multiple viewpoint images. They also used the multiple views to reduce self-occlusion as there where very prominent problems faced by singular camera gesture tracking.

Each hand position was tracked with a Kalman filter and the motion vectors were updated with image features in selected images that didn't include hand-hand occlusion. 3D hand postures were estimated with a small number of reliable image features. These features were extracted based on distance transformation, and they were robust against changes in hand shape and self-occlusion which resulted in a "best view" image which was selected for each hand by shape recognition. Fourier descriptors were used as features to describe postures and the system was proposed as a user interface device in virtual environments.

In 2003, Chen et al. [4] introduced a gesture recognition system to recognize dynamic hand gestures against a static background. The system consisted a real-time hand tracking and extraction, feature extraction, Hidden Markov Model training, and gesture recognition. They initially applied a real-time hand tracking and extraction algorithm to trace the moving hand and extract the hand region. Then, the hand region was used to extract Fourier Descriptors to characterize spatial features and the motion analysis to characterize the temporal features. They combined both spatial and temporal features of the input image sequence into the feature vector. This feature vector was used in a Hidden Markov model to recognize an input gestures which was separately scored against different Hidden Markov Models. The model with the highest score indicated the corresponding gesture. Experimental outcomes indicated that the recognition rate above 90% was possible with 20 different gestures. The next Figure 2.3 depicts the flow diagram of their hand tracking system.

In 2005, Premaratne et al. [30] developed a hand posture recognition scheme which provided 100% efficient for selected 10 gestures. These gestures were used to control consumer electronics control devices such as TV and DVD players. This mechanism used moment invariant features to individually represent hand gestures and an elaborate skin segmentation and morphological filtering mechanism to remove background and noisy regions. It used a hardware interface which used a computer parallel port to connect the recognized gestures to control other applications to perform different functions. The approach was invariant to translation, rotation and scale as well as to lighting variations, and the skin segmentation was very robust against many extreme skin colors. The classification stage of many gesture recognition systems use a single classification stage which provides strong classification when the selected features are unique and stand apart from other gestures. However, as it turns out, human gestures especially when using sign language, gestures vary incrementally in shape from one gesture to the other. This type of

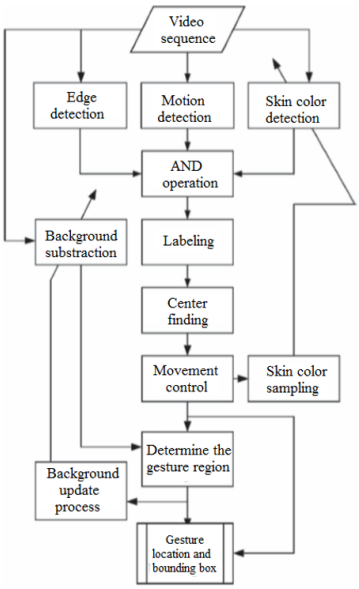

Figure 2.3.: The flow diagram of the hand tracking system based on Fourier descriptors and Hidden Markov Models. (Similar to [4])

scenario can be better served with a classifier with multiples stages where the rough initial classification is further classified using more details at a later step.

In 2006, Bing and Ejima proposed a multi-stage classifier for hand gesture recognition [5]. Their hand detector was based on a tree structure of boosted cascade of weak classifiers. The head of the tree forms the general hand detector and its only goal was to find all possible hand hypotheses in the image. Successful hypotheses were then passed onto the branches of the tree where specific cascades designed only to detect hands of a specific shape are used to determine the exact pose of the hand in the image. To build shape specific detectors the data set must be broken up into similar shapes that are for a specific shape. These kind of clusters were expected to contain sufficient variations for shapes to allow the classifier to generalize the process. They used an approach from their previous research to perform an intelligent selection of training images for training data. The framework of tree of hand detector is shown in Figure 2.4.

In order to detect a hand in an image, an exhaustive detection across all possible positions and scales were performed using a heuristic strategy. This approach was needed as the majority of the positions and scales would not contain the hand. The structure of the detector cascades resulted in many parameterizations that would be rejected in the first few layers of the top strong classifier, which required only a very small amount of computational cost. The results obtained with this system came up with a 99.5% correct recognition rate. These results proved that multilevel classifier could boost the classification confidence in gesture recognition.

However, from the work published by Appenrodt et al. [6], it can be deduced that the best gesture recognition can be offered by a camera system with depth information since stereo cameras could also suffer from occlusions. They used a multiple stereo camera setup with interesting results as illustrated in Figure 2.5:

in this system it was used colour information as well as depth information for detecting the hand of the user. Based on these 3D information of the scene, a representative point cloud of the hand was computed which is shown in blue points. They used an Iterative Closest Point algorithm to fit the hand model (red)

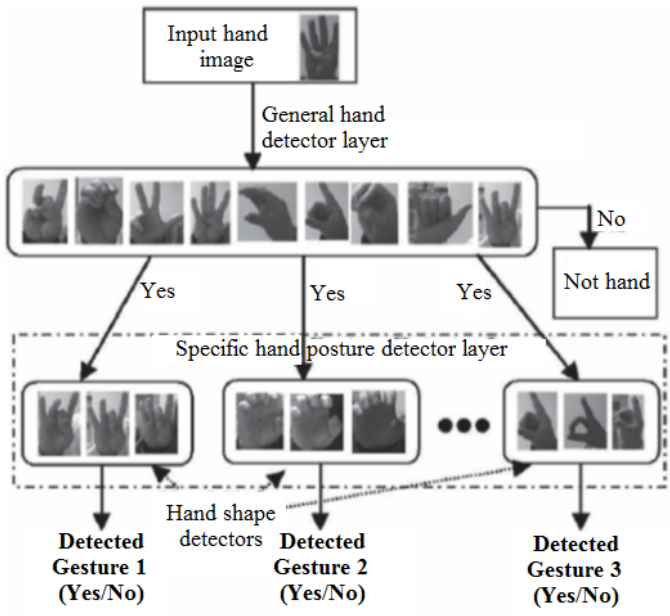

Figure 2.4.: The framework of a tree of hand detectors.(Similar to [5])

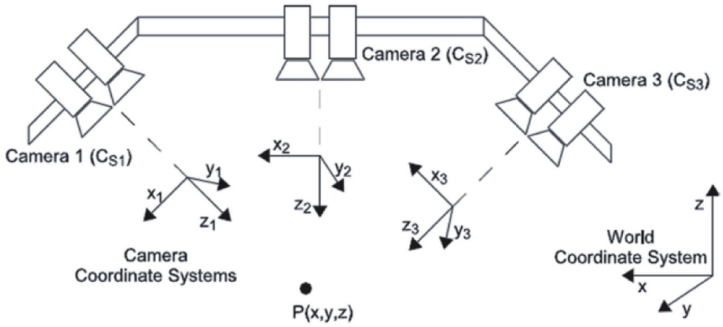

Figure 2.5.: Multi-stereo camera system. [6]

into the point as it can be seen in Figure 2.6:

2.3. Digital Image Processing

Before starting to recognize the gestures, a digital treatment phase has been taken place. This pre-processing part of the workflow is the process where it is taken an image and a modified version of it is produced in order to obtain specific information from it. An example is shown in Figure 2.7 where it has been isolated the interesting object from the background of the image.

This process can be divided into smaller processes in order to measure, interpret or classify the object as it can be seen in the Figure 2.8. The previous treatment that the image has to go through consist of 5 stages

1. Image acquisition: It is obtained an image of the object of interest.

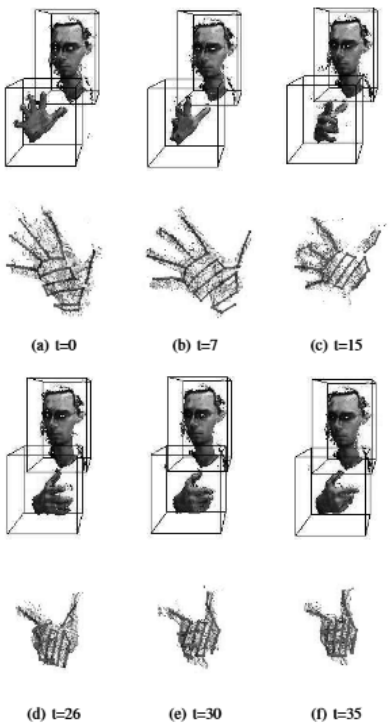

Figure 2.6.: Hand detection without self-occlusion using a multi-camera system.[6]

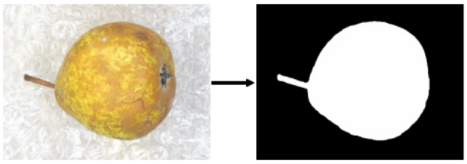

Figure 2.7.: Image segmentation. Detection of the object of interest.[1]

2. Restoration: there are applied digital filters in order to remove noise, increase the image contrast or brightness, etc. in order to improve the visualization of the obtained image. The result is more useful for a particular application.

3. Segmentation: The object of interest is separated from the background and from the rest of the objects.

    - Morphology: The mathematical morphology is a mathematical tool to extract image components that are useful in the representation and description of objects and improve the results of the Segmentation. These algorithms are focused on the shape of the object.

4. Description: Describe the object of interest through the measurement of certain attributes of the object. The parameters or measures that are extracted represent and characterize the object.

5. Recognition: To classify objects from the previous descriptors predefined. This means, to assign each object to a class which is a family of shapes (associated to objects) that share common characteristics. For example, the shape of a human body or the shape of a closed hand.

6. Interpretation: According to the values obtained in the measurement phase, is carried out an

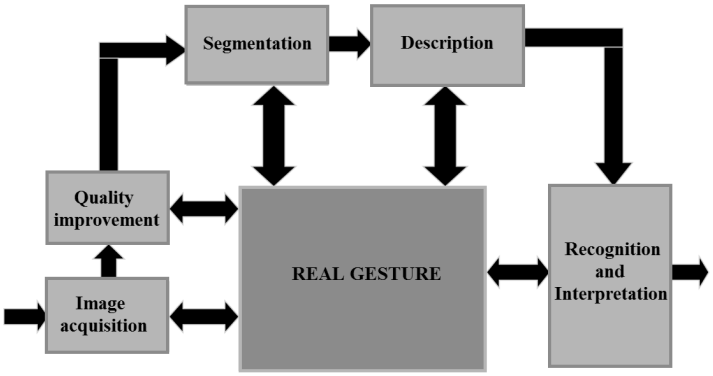

Figure 2.8.: Workflow of the image processing in order to detect and interpret gestures, postures or any other kind of objects. [1]

interpretation of the gesture.

This survey will focus mainly in the last two stages of this chain: Recognition and interpretation of gestures.

Chapter 3

Depth information extraction techniques

The ability to discern motion not only up-and-down and side-to-side but also front-to-back greatly expands the variety, richness and precision of the suite of gestures that a system can decode.

Depth information enables capabilities well beyond gesture recognition. The challenge in incorporating 3D vision and gesture recognition into technology has been obtaining this third "z" coordinate. Many visual recognition tasks are much easier with depth information. When processing a flat 2D image, pixels with similar colours that are near to each other might not belong to the same object. Instead, with 3D information pixels that correspond to locations physically near to each other tend to belong to the same object, irrespective of their colour. It has often been said that pattern recognition has been made artificially difficult because most systems rely on 2D data but once it is extracted the depth field many seemingly difficult tasks become much simpler.

There is no doubt the depth map is essential to the Kinect's working and it will be some time before the same tricks will be possible using standard video cameras.

In this section it will be treated three common technologies that can acquire 3D images, each with its own unique strengths and common use cases: stereoscopic vision, structured light pattern and time of flight.

3.1. Time of flight

This method is used in the second version of Kinect for Xbox One. Consists on a group of emitter and sensor that transmit a light pulse from the emitter to an object, in this case, the body and the receiver determines the distance of the measured object by calculating the travel time of the light pulse from the emitter to the body and back to the receiver in a pixel format [31]. Thus, matricial ToF cameras enable the acquisition of a distance-to-object measurement, for each pixel of its output data.

Time-of-flight systems are not scanners, as they do not measure point to point. Instead, they perceive the entire scene simultaneously to determine the 3D range image. With the measured coordinates of an object, a 3D image can be originated. Time-of-flight systems require a significant amount of processing, and embedded systems have only recently provided the amount of processing performance and bandwidth needed these systems.

3.1.1. Theory of operation

A 3D time -of-flight camera works by illuminating the scene with a modulated light source, and observing the reflected light. The phase shift between the illumination and the reflection is measured and translated to distance. Typically, the illumination is from a solid-state laser or a LED operating in

the near-infrared range (850nm) invisible to the human eyes. An imaging sensor designed to respond to the same spectrum receives the light and converts the photonic energy to electrical current. This means that the light entering the sensor has an ambient component and a reflected component. Distance (depth) information is only embedded in the reflected component. Therefore, high ambient component reduces the signal-to-noise ratio.

To detect phase shifts between the illumination and the reflection there are two different methods: Pulsed modulated or continuous-wave modulated sources, typically a sinusoid or square wave, although the square wave modulation is more common because it can be easily realized using digital circuits [32].

Pulsed modulated systems measure the time-of- flight directly and allow long-distance measurements. The arrival time must be detected very precisely. Needs very short light pulses with fast rise- and fall-times and with high optical power like lasers or laser diodes. Meanwhile, continuous-wave modulated systems measure the phase difference between the sent and received signals. Different shapes of signals are possible (sinusoidal, square waves, etc.). Cross-correlation between the received and sent signals allows phase estimation which is directly related to distance if the modulation frequency is known.

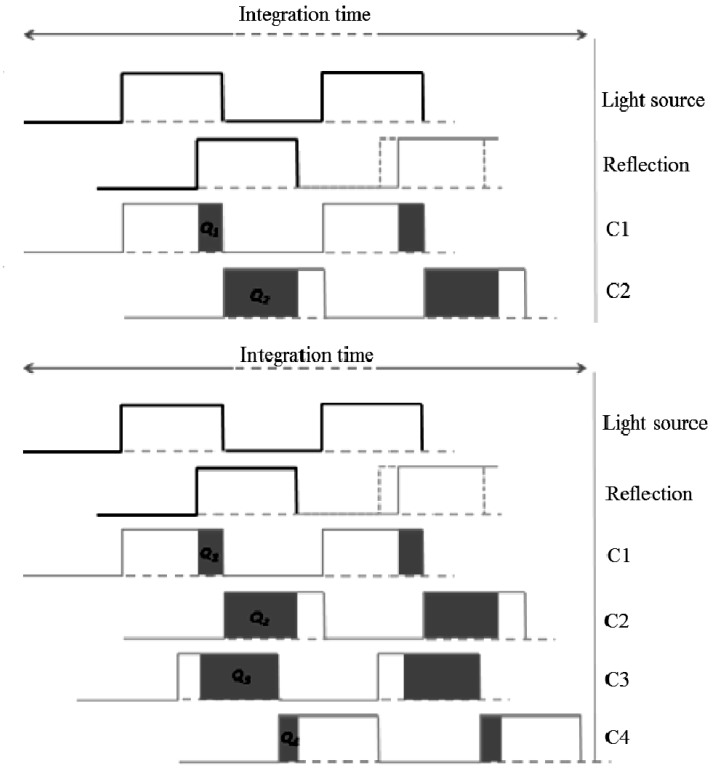

Figure 3.1.: Pulsed time-of-flight method (top), continuous-wave method (bottom). [7]

The pulsed method is straightforward. The light source illuminates for a brief period (Δ_t), and the reflected energy is sampled at every pixel, in parallel, using two out-of-phase windows, $C1$ and $C2$, with the same Δ_t. Electrical charges accumulated during these samples, $Q1$ and $Q2$, are measured and used to compute distance using the formula:

$$d = \frac{1}{2} c \Delta t \left(\frac{Q_2}{Q_1 + Q_2} \right)$$

In contrast, the continuous wave method takes multiple samples per measurement, with each sample phase-stepped by 90°, for a total of four samples. Using this technique, the phase angle between illumination

and reflection, ϕ, and the distance, d, can be calculated by:

$$\varphi = \arctan(\frac{Q_3 - Q_4}{Q_1 - Q_2})$$

$$d = \frac{c}{4\pi f}\varphi$$

In both equations c is the speed-of-light constant. It follows that the measured pixel intensity (A) and offset (B) can be computed by:

$$A = \frac{\sqrt{(Q_1 - Q_2)^2 + (Q_3 - Q_4)^2}}{2}$$

$$B = \frac{Q_1 + Q_2 + Q_3 + Q_4}{4}$$

A closer look at the continuous wave equations reveals that the terms, $(Q3\breve{}Q4)$ and $(Q1\breve{}Q2)$ reduces the effect of constant offset from the measurements. Furthermore, the quotient in the phase equation reduces the effects of constant gains from the distance measurements, such as system amplification and attenuation, or the reflected intensity. These are desirable properties. The reflected amplitude (A) and offset (B) do have an impact the depth measurement accuracy. The depth measurement variance can be approximated by:

$$\sigma = \frac{c}{4\sqrt{2}\pi f} \cdot \frac{\sqrt{A + B}}{c_d A}$$

The modulation contrast, c_d, describes how well the time-of-flight sensor separates and collects the photoelectrons. The reflected amplitude, A, is a function of the optical power. The offset, B, is a function of the ambient light and residual system offset. It can be appreciated that high amplitude, high modulation frequency and high modulation contrast will increase accuracy; while high offset can lead to saturation and reduce accuracy.

3.1.2. Point cloud

In time-of-flight sensors, distance is measured for every pixel in a 2D addressable array, resulting in a depth map which is a collection of 3D points (each point also known as a *voxel*). 2D representation of a depth map is a gray-scale image, as is illustrated in the depth map of Figure 3.2 where the brighter the intensity, the closer the voxel.

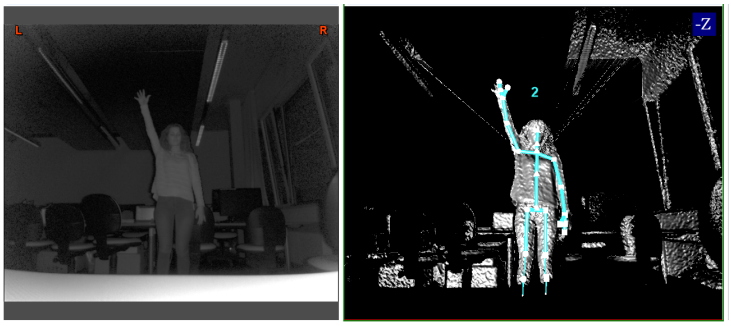

Figure 3.2.: Depth map of a human body.

Alternatively, a depth map can be rendered in a three-dimensional space as a collection of points, or *point-cloud*. The 3D points can be mathematically connected to form a mesh onto which a texture surface

can be mapped. If the texture is from a real-time color image of the same subject, a life-like 3D rendering of the subject will emerge, as is illustrated in Figure 3.3. One may be able to rotate the image to view different perspectives.

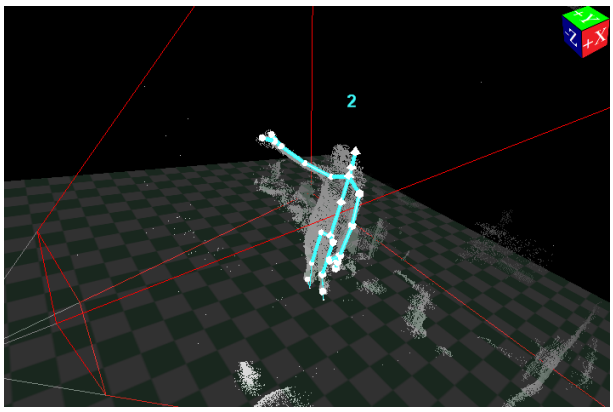

Figure 3.3.: Avatar formed from point cloud.

- Strengths

  1. Very simple and compact, comprising no moving parts and with incorporated illumination placed adjacent to the lens.

  2. Efficient due to the ease with which distance information is extracted when using time-of-flight cameras, only a small amount of processing power is used.

  3. Capacity to measure a complete scene with one shot with up to 100 frames per second making time-of-flight cameras much faster than their laser alternatives.

- Weaknesses

  1. At least two calibrated cameras required.

  2. Multiple computationally expensive steps.

  3. Dependence on scene illumination.

  4. Dependence on surface texturing

3.2. Structured light pattern

The first version of Kinect uses this kind of technology in which projected patterns can be used for measuring or scanning 3D objects. In this type of system, a structured light pattern is illuminated onto the subject and there is inspected the pattern distortion [33]. This light pattern can be created using a projection of laser light interference or through the use of projected images. Using cameras similar to a stereo vision system allows a structured light pattern system to obtain the 3D coordinates of the object.

Single 2D camera systems can also be used to measure the displacement of any single stripe and then the coordinates can be obtained through software analysis. Whichever system is used, these coordinates can then be used to create a digital 3D image of the shape. There are three different type of light that can be used: InfraRed Structured Light (IRSL), Imperceptible Structured Light (ISL) and Filtered Structured light (FSL).

3.2.1. Theory of operation

A structured light sensor is similar to a classical stereoscopic sensor with a camera replaced by a light source, as shown in Figure 3.4: normally, this light source is a laser or a slide projector. Projected pattern systems consists on a laser beam projecting a single dot all over the scene (Figure 3.4.a). Since only one point is projected, correspondence is made directly. However, it is required to scan along both axes. A second solution consists in projecting a light plane (that appears as a single slit on the object surface as depicted in Figure 3.4.b. In this case, scanning along one axis is still necessary. This is the technique commonly used for 3D scanning. In order to avoid a time consuming mechanical scanning, a last solution consists in projecting a multi-stripe pattern, a grid of multiple dots that is a bi-dimensional pattern (Figure 3.4.c and Figure 3.4.d).

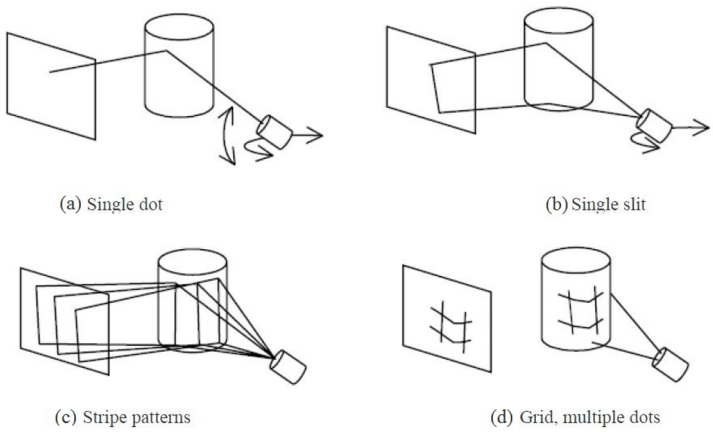

Figure 3.4.: Geometry of a structured light sensor. [8]

In IRSL an infrared laser beam is used to generate invisible patterns that can be a single dot, single line or a bi-dimensional pattern. The light is usually projected in the near-infrared (from 640nm to 2500nm). The scene is observed by a charge-coupled device (CCD) camera, because of the spectral sensibility of CCD, from 300nm to 1100nm (an infrared camera is not necessary).

In ISL, the imperceptible structured light sensors are composed by a unique light source and two cameras. The light source projects a light pattern followed by its complement (inverse pattern) onto the scene at high frequency, so that the resulting pattern is uniform. The first camera is synchronized with the projection of the first patterns and permits to reconstruct the scene thanks to the uniform light (as a result of pattern and complement projection) which permits to get a classical gray-level or colored image and process it.

In FSL, the light source is filtered so that only infrared structured light passes through. The light pattern can be projected by a laser source or a video projector. An IR filter, set in front of the light source, permits to "cancel" light below 750nm, 800nm, 850nm, etc. by acting as a high-pass filter.

3.2.2. Pattern codification

Coded structured light permits to increase the potentialities of structured light vision. [34]. there are three main strategies in order to encode patterns: time-multiplexing by projecting a sequence of different patterns onto objects; spatial codification when the coding of each pattern pixel depends on its neighbourhood; and direct codification when each pattern pixel carry its own code. Moreover, coding can be based on binary primitives (black or white), grey-level primitives or color primitives. ISL sensor guarantees a wide and complete codification. On the contrary, IRSL sensor cannot be coded at all (laser source prohibits easy codification). Due to the possible use of video-projector for FSL sensor, it is possible to encode the patterns through spatial binary codification or time-multiplexing codification. It is one of the main difference between IRSL and FSL: the latter offers modularity but with the disadvantage of harder implementation. In addition, it has to be mentioned that ISL and FSL sensor could be coded in an hybrid way., for example, by mixing visible code with invisible code, that is to say, by adding an invisible component to the classical codification primitives.

- Strengths [35]

    1. Can achieve very high spatial resolution since it uses a conventional imaging device.

    2. Do not require any special treatment at the sensor level.

    3. Fast system as it can scan multiple points or the entire field of view at once. Some systems enable the scanning of moving objects in real time.

    4. It avoids the correspondence problem of stereoscopic-based systems. This is because any disparity in the setup can be calibrated from the knowledge of distortion in the projected pattern.

- Weaknesses

    1. High cost of implementation.

    2. Difficulties providing the ideal beam geometry.

    3. Laser typical effects like speckle noise and the possible self interference with beam parts reflected from objects.

Chapter 4

Heuristic and machine learning methods for gesture recognition

In this survey has been exposed different gesture recognition techniques based, on one hand, on feature extraction, statistics and models (see Appendix B), on another hand, on machine learning (see Appendix A), and finally, on depth information extraction systems 3. As a result after having done the previous study of the different existing gesture recognition techniques, it can be made some conclusions about which methods are more efficient when talking about a gesture recognition application.

To sum up the benefits and the drawbacks of each method, it is exposed a summarized comparison of them using heuristic techniques or, otherwise, machine learning.

4.1. Heuristic-based gesture detection:

Heuristic is, basically, to shape an algorithm that is customized to describe a certain gesture. That algorithm is designed to recognize a single gesture or a single class of gestures relying on the context in which the gesture is performed using the skills and experience of a human to build that algorithm [36].

- **Benefits of using heuristics for gesture detection:**

    1. Less expensive than machine learning. It does not relay on external code basis or external libraries.

    2. No infrastructure costs.

    3. Helps reconstruct missing information.

    4. Reduces the expected states, or frames, of the gesture from the user's movements during detection and comparing them heuristically to expected values from a predetermined template.

- **Disadvantages of heuristics for gesture recognition:**

    1. Difficulty in describing algorithmically a gesture.

    2. Takes longer time to figure out an efficient algorithm that describes with high accuracy a complex gesture.

    3. Heuristics may produce results by themselves so they have to be commonly used in conjunction with other optimization algorithms to improve their efficiency

4. May not be the best of all the actual solutions, or it may simply approximate the exact solution.

4.2. Machine Learning for gesture detection:

Machine learning is, in brief, an algorithm developed to note changes in data and evolve in its design to accommodate the new findings using techniques that identify patterns within given datasets based on a likelihood matching with pre-existing patterns.This means that a machine learning approach involves the defining of gestures as an ordered set of gesture states based on a general expectation of what that state (gesture segment or pose) looks like mathematically in 3D space [37].

- **Benefits of using machine learning for gesture detection:**

1. Sophisticated pattern recognition. Along with noting relationships, can determine the type and quantify as well. This is not just happening with key, or even secondary variables, but on every relationship that takes part in the pattern.

2. Intelligent decisions along with the capability to note irrelevant data, and rank the relative importance of variables. Machine learning methods can make decisions either aided by a human or not. The solution can distinguish subclasses and make determinations on what data should be included and which should not with very little instruction.

3. Time saving since the total amount of time spent in training and testing the examples is much lower than trying to figure out an heuristic description method.

4. Gesture detection is created and handled as content, rather than being a time-consuming coding task.

5. Gestures can be quickly prototyped and evaluated before any code is written.

6. High accuracy for detecting gestures can be achieved—even in cases where skeletal data is very noisy, such as sideways poses.

7. By tagging data appropriately, perceived latency can be made very low.

8. The database size is independent of the amount of training data.

9. There are very few thresholds to tweak and maintain.

- **Disadvantages of machine learning for gesture recognition:**

1. As the model becomes more refined as much examples are given, there is needed to record the same gesture several times in different situations, speed, people, background, etc. The test of multiple iterations will produce a final version that delivers the highest level of accuracy.

2. The recorded gesture would be specific to the actor who recorded it and only those with a

similar frame would successfully have the gesture detected.

3. It is time consuming to tag data.

4. It is required a powerful PC for training.

5. It is required pretty much disk space for storing raw (.xrf) and processed (.xef) recordings.

Chapter 5

Kinect for gesture recognition: System overview

In the present work it is implemented an interaction framework where the user is able to interact with a virtual weather TV studio by performing the corresponding gestures in front of a Kinect sensor. Further, such system is compared with the Myo armband that detects the electrical activity from the arm muscles. For both sensors the accordingly SDK were used, to train and recognize gestures with machine learning.

This section covers an overview of the used devices and softwares that were needed to build up the Kinect system and how they are connected. The applications that runs on the specific machines will be described in the corresponding further sections.

5.1. System Architecture

The entire system is divided into four parts (see Figure. 5.3): The real space where the user, performs the gestures and receives visual feedback accompanied by a short sound each time he or she has performed correctly a gesture, the system has detected it and has triggered an animation. The Kinect sensor which provides depth, infrared, colour and skeleton data to the graphic engine.

In order to link both extremes, are run in parallel two bridging software. The Kinect2OSC which allows relevant skeletal data from the Kinect to be packaged into OSC signals to be used as the inputs to the OSC compatible graphic program. The second bridge, the KinectGesture2OSC, creates identification and individual OSC messages for each gesture learned by using machine learning techniques that was stored in the database. These OSC messages are received by the respective plug-in inserted in the graphic engine which, for this project, was the Viz Artist 3.8.2 software. Such software provides the user with the virtual TV weather studio visualisation and sends back audio and visual feedback to the real environment.

5.2. Kinect mechanism

Kinect hardware contains a RGB camera, a depth sensor and a four-microphone array, (see Figure 5.1) which are able to provide depth signals, RGB images, and audio signals simultaneously.

With respect to the software, several tools are available, allowing users to develop products for various applications. In this project have been used Visual Gesture Builder, Kinect Studio, Viz Artist and other tools and plugins that will be further explained with more detail.

For running the Kinect it was used a PC with a 64-bit and dual-core 3.1 GHz processor, an USB 3.0 controller dedicated to the Kinect, 4 GB of RAM, a graphic card that supported DirectX 11 and the operating system Windows 10.

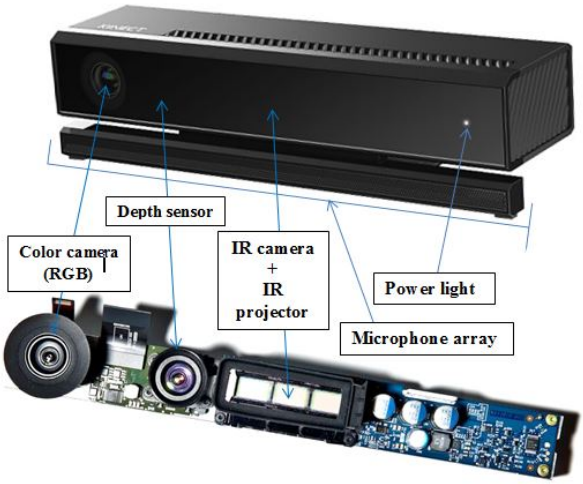

Figure 5.1.: Hardware configuration of Kinect [9, 10].

5.2.1. Sensing Hardware

Figure 5.1 shows the arrangement of a Kinect sensor, formed by an infrared projector, an infrared camera and a color camera. The Microsoft Kinect 2.0 sensor relies upon a novel image sensor that indirectly measures the time it takes for pulses of laser light to travel from a laser projector, to a target surface, and then back to an image sensor. This means, this version is therefore an instance of a time-of-flight light sensor.

Each component of the Kinect hardware is described below.

RGB camera

The color camera has a resolution of 1920 $\times$ 1080, and a field of view of 84,1 °(h) $\times$ 53,8 °(v). It delivers three basic color components of the video. The camera operates at 30Hz, and can offer images at 640$\times$480 pixels with 8-bit per channel. Kinect also has the option to produce higher resolution images, running at 10 frames at the resolution of 1280 $\times$ 1024 pixels.

Infrared camera

The infrared camera has a resolution of 512 x 424, and a field of view of 70,6 °(h) $\times$ 60,0 °(v). The two images, the one provided by the colour camera and the one coming from the infrared one, partially overlap in their centers, but do not fully contain each other. The colour camera captures a wider horizontal view of the scene than the infrared depth camera, which captures a slightly taller vertical view. These incomplete overlaps have practical implications when capturing point clouds: if only capturing points with associated RGB colour information, the point cloud will be cropped vertically compared to cases where no colour values are required.

As shown in Figure 5.2, three different output streams arise from the two lenses of the Kinect v2 device: infrared data and depthmaps come from one lens and have the same resolution. The depthmaps are

2D images 16 bits encoded in which measurement information is stored for each pixel. The colour images of higher resolution come from the second lens. The point cloud is calculated thanks to the depthmap, because of the distance measurements it contains.The result is a list of (x, y, z) coordinates that can be displayed as a point cloud.

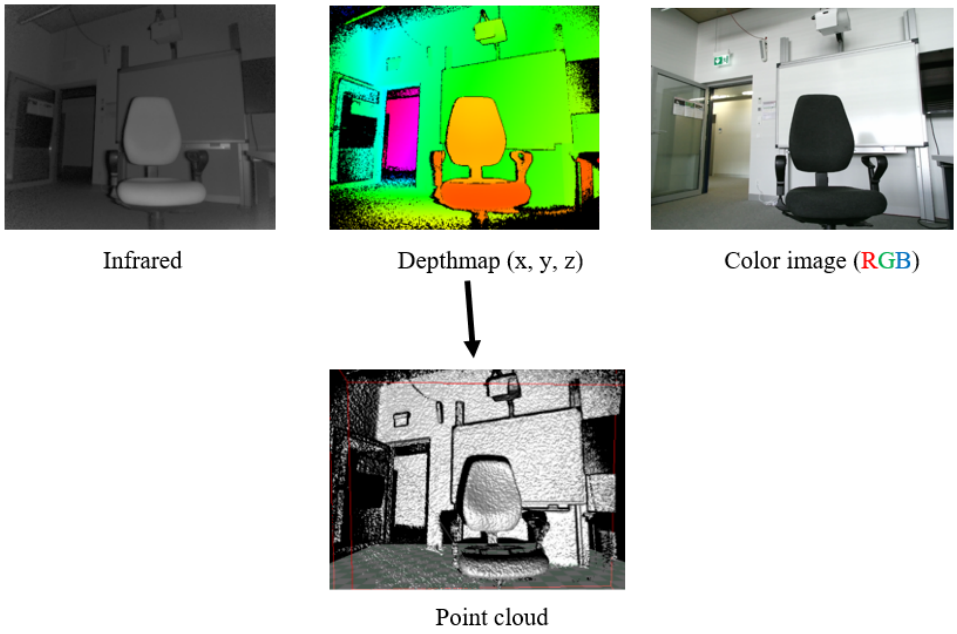

Figure 5.2.: Schematic representation of the output data of Kinect v2.

5.3. Bridging

Once the database with the gestures is trained with sufficient number of example frames, it must be connected to the virtual TV studio set. For doing so, it has to be created a bridging software which has to submit the corresponding commands to the virtual set. These commands are sent as OSC messages. The database with the trained and tested gestures was loaded into the KinectGesture2OSC software. The KinectGesture2OSC software receives the skeleton information from the Kinect and performs two simultaneous tasks:

Allows the user to visualize the skeleton data that is provided by the sensor in real time as well as the hand state. It shows a colour circle around the four joints that compose each hand: green for an open hand, red for a close state, blue for a pointing pose and grey for an unknown hand state.

On another hand, this bridging software generates a specific and unique OSC message for each of the gestures that were stored in the database. In the case of the discrete gestures, just a boolean value is sent, instead, for the continuous performed gestures it sends the exact progress value of them. In this way, each time the user performs one of the 8 gestures, the bridging software generates the corresponding OSC message and sends it to the OSCPugin that is included in the virtual scene of the graphics engine.

KinectGesture2OSC was coded using C# language and taking as basis the example *Discrete Gesture Basics-WPF* available in the Kinect SDK for Windows v2.0.[1] which was adapted and completed to be

[1] `https://developer.microsoft.com/en-us/windows/kinect/tools`

able to detect not only discrete gestures but also continuous ones.

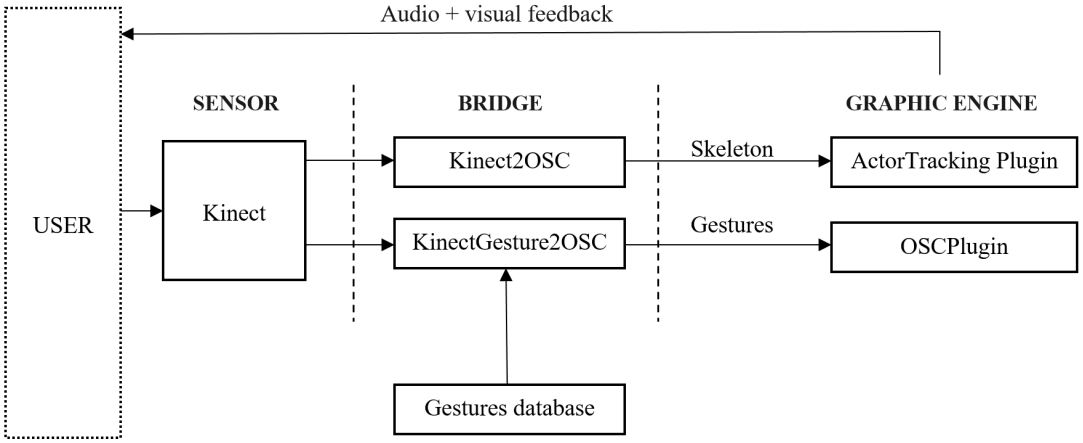

Figure 5.3.: Interactive Kinect framework.

Chapter 6

Gestures database building

With the purpose of controlling the different elements of the virtual set it was implemented a database with all gestures that were needed. This section provides with an overview of all the steps that were followed in order to build it up.

6.1. Implemented gestures

In the Table 6.1 are exposed which gestures where implemented in order to interact with the virtual scene.

INTERACTION	**GESTURE**
Starting the application	Wave
Selection of the *Temperatur* button	PointUp
Selection of the *Video* button	PointMiddle
Selection of the *Morgen* button	PointDown
Scrolling up the map	ScrollUp + ScrollProgress
Scrolling down the map	ScrollDown + ScrollProgress
Scrolling right the map	ScrollRight + ScrollSideProgress
Scrolling left the map	ScrollLeft + ScrollSideProgress
Zooming in the map	ZoomIn + ZoomProgress
Zooming out the map	ZoomOut + ZoomProgress

Table 6.1.: Gestures that were implemented in order to interact in the indicated ways.

The gestures consisted as follows:

- *Start*: Moving the right hand from the left side of the head to the right drawing a straight horizontal line.
- *PointUp*: Pointing with the right hand over the user's head while opening the left hand.
- *PointMiddle*: Pointing with the right hand between the eyes level and the navel height while opening the left hand.
- *PointDown*: Pointing with the right below the navel height while opening the left hand.

The previous gestures are discrete (boolean values) which means that the simple fact of performing them, triggers the corresponding animation.

- *ScrollUp*: Moving the opened right hand upwards nearby the right side of the body.
- *ScrollDown*: Moving the opened right hand downwards nearby the right side of the body.
- *ScrollRight*: Moving the opened right hand to the right nearby the right side of the head frawing a straight line.
- *ScrollLeft*: Moving the opened right hand to the left nearby the left side of the head drawing a straight line.
- *ZoomIn*: Moving both closed hands of the centre (at the eyes' level) outside.
- *ZoomOut*: Moving both closed hands from the outside to the centre.

These gestures are continuous (floating point values) which means that it can be obtained the exact progress value of the gesture. This value is used to interact with the map in a more accurate level.

6.2. Collection of relevant data

It was recorded the clips that contained each gesture separately by using the tool Kinect Studio[1] in order to get the Raw IR stream provided by the Kinect sensor (see Figure 6.1). By doing so, it was recorded the depth, IR, and body frame data streams from the sensor array. It was included a diverse variety of situations, clothes, distances to the sensor, backgrounds, ways of performing the same gesture and contexts where a certain gesture occurred in order to generalize it. Also, it was recorded similar gestures that might be confused by the classifier in order to train the system also with negative examples.

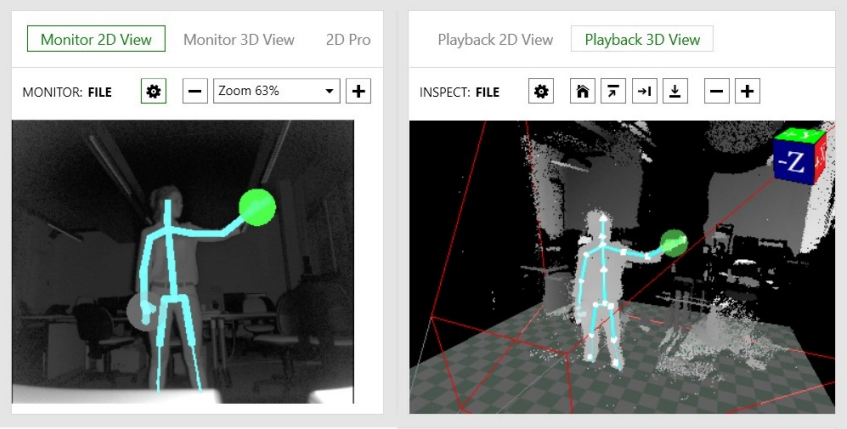

Figure 6.1.: Left: 2D infrared view of the scene with the skeleton . Right: 3D depth view of the same scene as the left one.

Each people was recorded individually since people are easily influenced in the way they perform a gesture when they see how others perform the same gesture or when other people watch how they perform it. It was told them to perform the gesture, but not exactly how, in order to have recordings of the widest possible range of ways that people perform relevant gestures in their own unique ways. As the raw data

[1] https://msdn.microsoft.com/de-de/library/dn785306.aspx

streams from the recorded gestures that are stored in an eXtended Raw File (.xrf) ,that is processed IR data, depth data, and skeletal data, were not directly usable by other application, it was needed to convert them into eXtended Event Files (.xef) by using the KSConvert tool provided by the Kinect for Windows SDK 2.0.

6.3. Training

Once the collection of data has been made, the recorded data was divided into two parts: one for building/training and another for analysing/testing the gestures since the accuracy of the machine-learning algorithms cannot be measured by testing it on the same data that it has been trained with. Approximately the 66% of the data was used for training and the 33% for testing. For this approach, it was used the Visual Gesture Builder (VGB) tool [38, 11].

As the raw data streams from the recorded gestures that are stored in an eXtended Raw File (.xrf) are not directly usable by other applications, it was needed to convert them previously into eXtended Event Files (.xef) by using the KSConvert tool provided by the Kinect for Windows SDK 2.0. Then, it was built a database with the 8 gestures that are used in the application using the VGB. Such tool provides a data-driven solution to gesture detection through machine learning. This means that gesture detection is turned into a task of content creation, rather than code writing. A small gesture database is built using VGB, and using the database has very low run-time costs in terms of memory overhead and CPU processing. Such database was divided into two groups: discrete gestures and continuous ones (see Figure 6.2); and each of them were analysed using different machine learning algorithms.

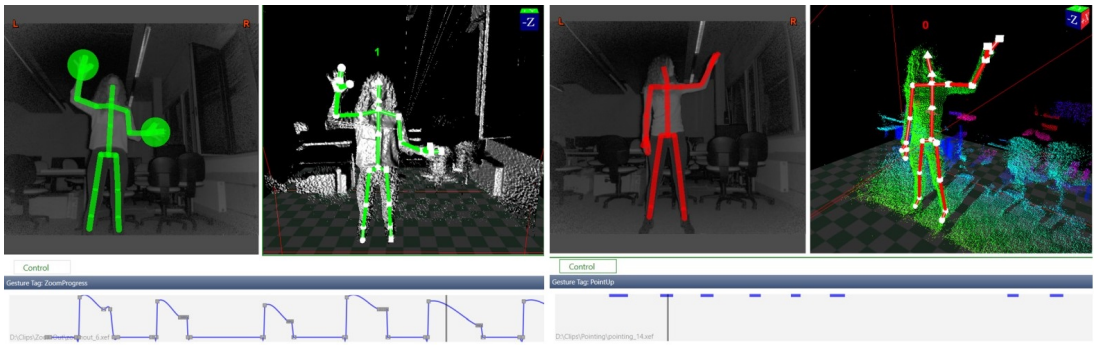

Figure 6.2.: Tagging process of the discrete *PointUp gesture* (left) and the continuous *ZoomOut* gesture (right) using the Visual Gesture Builder tool.

6.3.1. Machine Learning

This means the ability of an algorithm to automatically learn to recognize complex patterns in data. A computer usually does this by learning from empirical data examples, and the result is that it can classify data that it has not yet observed in an unsupervised learning process. There are many different approaches to machine learning, such as boosting algorithms, neural networks, decision trees, support vector machines (SVM), Bayesian networks, and so on. In this project, the Adaptive boosting algorithm [39] was used to train discrete gestures and the Randomized Forest Regression algorithm [40] was needed for the continuous gestures.

Adaptive boosting algorithm

The Adaptive Boosting machine learning algorithm was used to determine when the user was performing a certain gesture. Such detection technology produces a binary or discrete result whether the actor is performing the gesture or not. During the training process it accepted input tags, boolean values, which marked the occurrence of a gesture. This marking or tagging is used to evaluate whether or not a gesture is actively happening and determines the confidence value of the event. Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules in order to build up a strong classifier. Intuitively, for a learned classifier to be effective and accurate in its predictions, it should meet three conditions:

- It should have been trained on “enough” training examples.
- It should provide a good fit to those training examples (low training error).
- It should be “simple.”

This conditions where always present in the training phase and the more complex gestures such as Zooming where splitted into simpler parts: *ZoomingIn* and *ZoomingOut* segments (when the user was actively moving) , *MaxIn* and *MaxOut* segments (when the user performed the maximum level of the zooming gesture), and *keepLevelZoom* (when the actor stopped the gesture and the map should keep the scale in that point). All of them belonged to the gesture *"Zooming"* but such a complex continuous gesture had to be devided into discrete segments in order to raise the accuracy of the system.

For such training approach, the input parameters that were provided to the algorithm are described below: The accuracy level value controlled how accurate the results were, but also affected the training time. The higher the accuracy, the longer the training time was. The algorithm can potentially generate tens of thousands of weak classifiers. Using all of them increased accuracy, but at the higher CPU cost on the computer. For example, the CPU cost for 1000 weak classifiers is only of 25 microseconds https://msdn.microsoft.com/en-us/library/dn785522.aspx and the results have more than adequate accuracy.

It was also needed to use a filter to be applied on the raw per frame results since the results of the algorithm are based on a frame, not on a gesture. The Adaptive Boosting algorithm that was used in VGB, provides a simple low latency filter and is a simple sliding window of N frames, summing up the results and comparing it against a threshold value. The number of frames can be seen as a frequency and the threshold can be seen as amplitude.

For this application where the gestures that are used only need the algorithm to focus on the arm movements, it was ignored the lower-body joints. This reduced the amount of input data and also makes the gestures able to be performed in both positions, seated or standing.

Also was used the hand state as an input parameter. Kinect can differentiate between three different kind of hand states: open, close, and pointing. In this way, the gestures *Start*, *ScrollUp* and *ScrollDown* were trained with an open-hand state; *PointUp*, *PointDown* and *PointMiddle* with a pointing-hand position; and the gestures *ZoomingIn* and *ZoomingOut* with a close hand. This was made to make the system differentiate better between similar gestures such as *PointingUp* and *ScrollingUp* since the first one is performed with a pointing position and the last one with an open hand.

In training using such algorithm it was also important to include not only positive examples of the gesture but also negative ones to make the algorithm also learn which movements didn't belong to the certain gesture that was being trained. By doing so, it was needed to include in the training set some gestures that might look similar to the one that it was currently being trained and the algorithm could confuse.

Randomized forest regression algorithm

Such detection technology produced an analog or continuous result. The Random Forest Regression (RFR) machine learning algorithm determined the progress of a gesture performed by a user. It used tagging models that were represented by analog signals that occur during the gesture. Examples of these models could be an abstract concept such as the "progress" of a gesture or a quantifiable value such as the angle of an arm in scrolling gesture. Valid ranges for the RFR tags were based on the model that the analog signal represented. The RFR is a context-based detector, that means that the progress detection is only valid when a user is performing a gesture. The progress detection could be enabled as follows:

1. When the application was aware that the user was performing a gesture, it enabled the detection and used it to aid in rendering a simulation.

2. Progress detection was enabled when a discrete gesture (Adaptive boost algorithm) was detected.

For doing so, the continuous gestures were divided into discrete simpler segments and it was tagged the Discrete version of the gesture, since it was needed to use a Discrete gesture to give context to the Continuous gesture result (which always reports a progress value, even when no gesture is occurring). When tagging the Discrete gesture, it was included all frames where the continuous motion was actively occurring, but there was excluded any preparatory or recovery motions as this could confuse the trainer. In this way, only if the Discrete gesture was set up to 1, the Continuous detector started working. For such training approach, the input parameters that were provided to the RFR algorithm are described below and showed in Figure 6.3:

Name	Value	Type
Use Rotation Variant Feature	False	BOOL
Number of Trees	50	INT
Max Tree Depth	50	INT
Cluster Threshold	0.3	FLOAT
Weight Factor	0.2	FLOAT

Figure 6.3.: Screenshot of the input parameters that it can be used when tagging a gesture.

The rotation variation feature makes the gesture detector take into account the user's global rotation. In this case, the detection produces a stronger signal that is less robust to orientation change. Instead, if it's not used this parameter, the gesture detection tries to ignore global rotation as much as possible. For example, a gesture with the user head on and the same gesture with the user at 45^{o} returns a similar signal. The number of trees affects the memory, and disk space. Larger values give better results at the cost of more memory and disk space, and also increases run time CPU use for searching through these trees but the accuracy of the detector will improve. Also was possible to determine the maximum depth of the trees in order to control the maximal complexity of a decision tree. values allow a tree to grow deeper for complex scenarios, at the cost of more memory space and CPU use.

The cluster threshold determines the level of changes that can occur from frame to frame. The lower this value is, the less change can occur and more localized will be the searching, which could be more accurate if the search is performed in the correct neighbourhood when no good match could be found. This could produce better overall results, however, there will be more misdetections. The last parameter takes into account the weight of the output from the previous frame that will affect the current frame's output.

6.4. Analysis of the gestures database

During the building process, the appropriate machine learning algorithms processed the tagged data. This created a gesture database which afterwards was loaded into the KinectGesture2OSC at run time.

The Figure 6.4 represents the data that was used to build and analyse the discrete *ScrollDown* gesture. In this case, the hand state and only the upper parts of the body are taken into account and are excluded all the joints that belong to the lower part.

```
            Data Streams:
                  Use Hands Data: Yes
                  Use Skeleton Data: Yes
            Joints:
                  Ignore Left Arm: Yes
                  Ignore Right Arm: No
                  Ignore Lower Body: Yes
                  Effective Joint Mask: 0xfffffffff911f1f
                  Effective Hand Mask: 0x2
                  Excluded: ElbowLeft, WristLeft, HandLeft, KneeLeft, AnkleLeft, FootLeft,
KneeRight, AnkleRight, FootRight, HandTipLeft, ThumbLeft
                  Included: SpineBase, SpineMid, Neck, Head, ShoulderLeft, ShoulderRight,
ElbowRight, WristRight, HandRight, HipLeft, HipRight, SpineShoulder, HandTipRight, ThumbRight
```

Figure 6.4.: Data that was used to build and analyse the discrete *ScrollDown* gesture.

While training and building the database, the system takes into account all the positive frames that have been tagged as positive examples of the gesture and also the negative ones that represent the frames that do not belong to it as shown in Figure 6.5

```
Duration: 0 hours, 7 minutes, 16 seconds
Num Labeled Examples: 85216
Ratio positive to negative labeled examples: 1.0 : 44.47279
Total Num Gestures: 1638
Num Gestures Labeled as Scroll_Down: 44
Ratio Scroll_Down to all other gestures: 1.0 : 36.22727

Done

Generating labeled Examples...Done. Took (16) seconds
Feeding Examples...
    Positive( 1874 ), Negative( 83342 )

Step 1 of 4: Loading Labeled Examples
Examples Loaded: Positive(1874), Negative(83342)
Done
```

Figure 6.5.: Generation of examples for the training process with positive and negative tagging of the data.

After the examples have been loaded, it must be generated a pool of weak classifiers. The Figure 6.6. Represents the weak classifiers used to characterize and distinguish the discrete *ScrollDown* gesture. In this case, 38 different features are taken into account while looking for a repetitive pattern that was located in all the positive examples of the gesture. Each feature had attached a certain number of weak classifiers depending on how important they were in the description of the gesture.

```
Step 2 of 4: Generating a Pool of Weak Classifiers
       Using Hands data: Yes
       Using Skeleton data : Yes
        1/38 - Feature: DiffPositionX (70)
        2/38 - Feature: DiffPositionY (93)
        3/38 - Feature: DiffPositionZ (114)
        4/38 - Feature: Angles (771)
        5/38 - Feature: TimeSpaceAngles (79)
        6/38 - Feature: Speed (105)
        7/38 - Feature: VelocityX (86)
        8/38 - Feature: VelocityY (109)
        9/38 - Feature: VelocityZ (102)
       10/38 - Feature: AngleVelocity (212)
       11/38 - Feature: AngleAcceleration (152)
       12/38 - Feature: MuscleForceX (41)
       13/38 - Feature: MuscleForceY (87)
       14/38 - Feature: MuscleForceZ (65)
       15/38 - Feature: MuscleTorqueX (248)
       16/38 - Feature: MuscleTorqueY (50)
       17/38 - Feature: MuscleTorqueZ (188)
       18/38 - Feature: MusclePower (100)
       19/38 - Feature: DiffMuscleForceX (131)
       20/38 - Feature: DiffMuscleForceY (155)
       21/38 - Feature: DiffMuscleForceZ (161)
       22/38 - Feature: VelocityX^2 (32)
       23/38 - Feature: VelocityY^2 (44)
       24/38 - Feature: VelocityZ^2 (55)
       25/38 - Feature: Speed^2 (79)
       26/38 - Feature: Acceleration (68)
       27/38 - Feature: AccelerationX (37)
       28/38 - Feature: AccelerationY (34)
       29/38 - Feature: AccelerationZ (42)
       30/38 - Feature: BoneLengthChanges (172)
       31/38 - Feature: HandValueRaw (196)
       32/38 - Feature: HandValueMultiClass (167)
       33/38 - Feature: HandDifferenceRawBest (152)
       34/38 - Feature: HandDifferenceMultiClassBest (132)
       35/38 - Feature: HandDifferenceRaw (162)
       36/38 - Feature: HandDifferenceMultiClass (200)
       37/38 - Feature: RefinementWrist (17)
       38/38 - Feature: RefinmentHand (37)
       Total Classifiers Combined: 4745
       Num weak classifiers generated: 4745
       Duration: 8 minutes, 52 seconds
Done
```

Figure 6.6.: Generation of a pool of weak classifiers.

Once the weak classifiers are generated depending on the importance weight of each feature, it was built a strong classifier by combining the multiple weak ones with a higher accuracy when taking decisions. For doing so, the Adaptive boosting algorithm, was applied.

```
Step 3 of 4: Training Strong Classifier
       -Evaluating classifier data for each example skeleton frame...
Done
       -Running AdaBoost using 8 (of 8 available) hardware threads...
Done
       Num weak classifiers: 1000
       Duration: 0 minutes, 23 seconds
```

Figure 6.7.: Evaluating the classifiers using the Adaptive Boost algorithm.

Then, (see Figure 6.8) it is applied a filter to analyse a short range of frames at once instead of analysing the whole clip. Also, it is determined a detection threshold. Lower values of this threshold could increase true positives, at the risk of increasing false positives. Instead, higher values could decrease false positives, at the risk of decreasing true positives.

```
Step 4 of 4: Optimizing detection parameters
       -Generate matrix of test results using different detection parameters...
       Filtering 9 frames using detection threshold 0.037000
       Duration: 0 minutes, 4 seconds
Done
```

Figure 6.8.: Generation of the classifiers matrix and filtering process.

In this case, in Figure 6.9 can be seen in which level has intervened each weak classifier in the final classification of the *ScrollDown* gesture. This information includes details of features considered by the algorithm including a top ten features list. The features are very enlightening and give an insight into each particular gesture. In each row it can be seen the "fValue" which is a ratio of two variables: F = F1/F2, where F1 is the variability between groups and F2 is the variability within each group [41]. This fValue reveals the discriminative power of each feature independently from others. The "alpha" value is the weight applied to each classifier as determined by the adaptive boost algorithm. So the final output was just a linear combination of all of the weak classifiers, and then it was made the final decision with all of them. The classifiers were trained one at a time. After each classifier was trained, it was updated the probabilities of each of the training examples appearing in the training set for the next classifier. The first classifier was trained with equal probability given to all training examples. After it was trained, it is computed the output weight (alpha) for that classifier.

```
Top 10 contributing weak classifiers:
      HandDifferenceMultiClass( Right, CLOSED, LASSO ) using inferred joints, fValue >= 0.650000, alpha = 1.899626
      HandDifferenceMultiClass( Left, UNKNOWN, LASSO ) rejecting inferred joints, fValue >= 0.520000, alpha = 0.545869
      Angles( HandRight, WristRight, ElbowRight ) rejecting inferred joints, fValue >= 148.000000, alpha = 0.527014
      HandDifferenceMultiClass( Right, CLOSED, LASSO ) rejecting inferred joints, fValue >= 0.530000, alpha = 0.482087
      Angles( ShoulderLeft, SpineShoulder, ShoulderRight ) rejecting inferred joints, fValue >= 158.000000, alpha = 0.465442
      Angles( HandLeft, WristLeft, ElbowLeft ) rejecting inferred joints, fValue >= 152.000000, alpha = 0.455160
      MuscleTorqueX( HipRight ) rejecting inferred joints, fValue >= 0.299997, alpha = 0.404051
      HandValueRaw( Left, CLOSED ) rejecting inferred joints, fValue >= 0.520000, alpha = 0.374547
      HandDifferenceMultiClass( Right, OPEN, LASSO ) rejecting inferred joints, fValue >= 0.490000, alpha = 0.340463
      MuscleTorqueZ( ShoulderLeft ) rejecting inferred joints, fValue >= 2.799997, alpha = 0.338100

Done

Training...Done. Took (23) seconds
```

Figure 6.9.: Generation of the classifiers matrix and filtering process.

6.5. Testing

When creating a gesture prototype, a small set of training clips (between 10 and 20) were sufficient, but not all the gestures were trained with the same amount of examples. In order to know when it was established enough example data, it was needed to determine the amount of false positives and false negatives when testing the database. An example of a false positive is when a user performs gesture B, but gesture A is detected. Instead, a false negative is when a user performs gesture A, but detection fails to identify that gesture A has been performed. The number of these two errors are two different values that needed to be interpreted separately since the one is not the inverse of the other. The results of analysing the database provided the count of errors from false positives and false negatives as shown in Figure 6.10.

```
Testing on Training Data
    Raw Per Frame Results:
       % Accuracy True Positives: 99.306297 % (1861/1874)
       % Error False Positives: 0.011999 % (10/83338)
    Filtered Per Gesture Results:
       % Accuracy True Positives: 100.000000 % (28/28)
       % Error False Positives: 7.142857 % (2/28)
```

Figure 6.10.: True and false positives rate.

The graphic in Figure6.11.represents how these false positives and false negatives values change as new training data is being added to the training set. As it can be seen, at very few number of training examples, the error rates are high, but as more training examples are added, the error rates decrease notably. However, at a certain point, the error rates start to stay the stable, regardless of how many more training examples are added. When both values for false positives and false negatives start to flatten out at a low error rate, it is safe to assume that there is enough data in the training set.

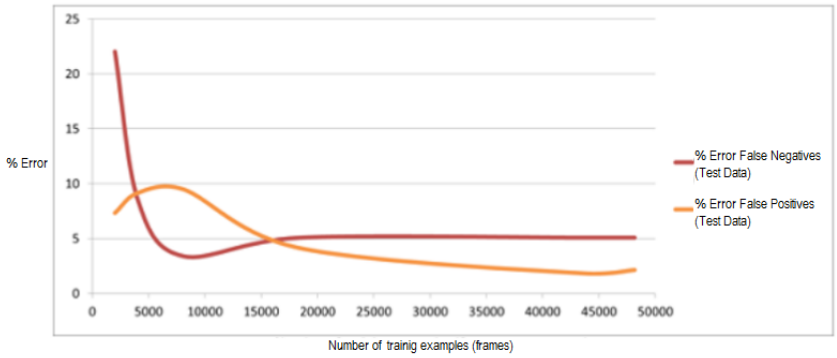

Figure 6.11.: True and false positives rate. (Similar to [11])

When both values for false positives and false negatives start to flatten out at a low error rate, it is safe to assume that there is enough data in the training set.

Chapter 7

Virtual set

A virtual weather studio scene was designed with the graphic engine Viz Artist 3.8.2 [1]. The virtual scene consists of a ground floor, a window row located in the background, a menu composed by three selective buttons: *Temperatur* on the top, *Video* in the middle and *Morgen* on the bottom and four displays that can be controlled by the previous buttons. Each display is attached to each button in such a way that when the user performs the corresponding gesture, the related button is selected and then the attached display is showed off. A visual example of the virtual set that it was designed is shown in Figure 7.1.

Figure 7.1.: The virtual weather studio used for testing the gesture control.[12]

It was used two OSC plugins within the graphic engine to establish communication between the sensor and the virtual scene. The ActorTracking Plugin receives OSC messages containing the skeleton data from the Kinect2OSC bridging software. Instead, the OSCPlugin received information from the KinectGesture2OSC software which sends the corresponding OSC messages each time the user performs a gesture within the predefined gestures database.

Also, it was used the Audio Plugin included in Viz Artist in order to release audio feedback to the user.

As it was explained in section 6.1, the discrete gestures triggered the respective animations, instead, doing a continuous gesture allows the user to control exactly the map in real time at a certain level according to the performed gesture.

[1] http://www.vizrt.com/products/viz-artist/

7.1. Open Sound Control

Open Sound Control (OSC) is a network protocol for communication between computers, synthesizers and other multimedia devices. It has been optimized for modern networks and is mainly designed to share data in real time over a network.

OSC is a transport-independent, message-based protocol. This means that the OSC messages do not require to be sent in any particular way. The details for any given situation will vary depending on how OSC is being used. In the Kinect case, OSC supports a client/server architecture. Clients are devices that send OSC data packets to the server. This can be a physical device, a software or a background program. All OSC data packets a will be referred to a server. In the case of the Kinect system, the OSC client, the KinectGesture2OSC generates an OSC message for each gesture that is being performed and it sends its value to the OSC server, in this case, the VizArtist through the OSC Plugin.

For such approach is needed a known IP address and a dedicated port for creating the connection. Boolean values are sent in the case of discrete gestures and float values if continuous gestures have been performed. For example, the KinectGesture2OSC creates the OSC connection by defining an IP address and a port:

```
IPEndPoint myDetector = new IPEndPoint (10.50.17.247, 8000);
IPEndPoint vizArtist = new IPEndPoint (10.253.185.135, 7000);
```

By doing so, it is defined that the device where KinectGesture2OSC is running has the IP address `10.50.17.247` and the port that is being used by the client to send the OSC messages is `8000` . If the user performs, for example, the gesture *PointUp*, the client generates the OSC message `vizArtist/PointUp/1.0f` , this means that the boolean value 1.0f is being sent to the port `7000` of the device where the OSC server is being run which has the IP address `10.253.185.135` .

7.2. Animations

Under animation is meant preprogrammed sequences of virtual objects where there are a fixed start and end points. What is happening between these points are defined by keyframes. The animations are triggered by the user each time that performs the corresponding gesture so the object initializes a smooth transition between keyframes.

There were created seven different animations: Starting, selection of the highest button, showing the first display, selection of the middle button, showing the second display, selection of the third button, showing the third display.

The starting animation is triggered by the *Wave* gesture and consist on the appearance of the *Menue* (the three buttons on the right side), followed by the showing of the initial display and the rotation of the window set.

When the user performs the gesture *PointUp*, there are triggered two animations: the highest button is

shown as selected by looking bigger and the first display is shown on the screen. In the same way, the other two buttons can be selected and can be triggered the corresponding animations.

7.3. Tracking

Unlike the pre-established animations, permanent data is sent to the virtual objects when continuous gestures are being performed. The exact value of the continuous gesture can be used in order to transmit the movement of the arm to the virtual object. A simple surface thereby moves in the direction in which shows the arm. In the example of the weather TV studio, this is used to move the video display by zooming and scrolling the map. There were created six different tracking animations: zooming in the video, zooming out the video, scrolling up, scrolling down, scrolling to the right and scrolling to the left. Such tracking animations were controlled by the corresponding continuous gestures.

7.4. Feedback for gesture recognition

Some of the accuracy limitations of gesture recognition systems can be overcome by providing feedback to the user, making the recognition process less opaque. Many conventional systems can provide sound, vibrational or visual feedback on the completion of a gesture in such a way that the user can identify any possible errors. However, dynamic, continuous feedback during the gesture allows the user to respond to the system in real-time. This avoids repetition and can help the user to understand where and when errors or deviations are happening while performing the gesture. For these reasons, providing feedback in a gesture recognition system is an important matter that requires attention.

7.4.1. Visual feedback

Visual feedback is a visual output from a system that allows the user to interact better with the system. An example is when interacting with a TV studio scene in a green screen, the images on the monitor are the visual feedback of the system [42]. The use of visual feedback to show users when their interactions are detected, interpreted, and handled. Visual feedback can help users by encouraging interaction. It indicates the success of an interaction, which improves the user's sense of control. It also relays system status and reduces errors.

Many researches have been made in order to examine the effectiviness of using visual feedback (among other types of feedback) for various modalities such as 3D gestures [43], and hand tracking gestures [44]. The conclusions of them came up with the idea that using a visual feedback is totally necessary to allow users to understand that their input has had the desired effect and instead of having the computer interpret a gesture and act on it, the user may be able to see exactly how the computer is estimating the gesture.This allows the user to correct errors that may appear in the body tracking . This approach may be suited for applications, such as gesture recognition and 3D virtual reality applications.

In the present project, the user receives as visual feedback the vision of the virtual TV weather studio with its interactive animations in an independent monitor run by Viz Artist, as well as the skeleton of his/her own body run by KinectGesture2OSC software.

7.4.2. Audio feedback

A simple model for enhancing user performance while using gesture recognition applications is to simply augment the user's perception with audio. This involves sonfiying the aspects of the gesture state to physical movements such as acceleration or velocity. Also the user can be provided with a specific kind of sound each time he or she accomplishes a specific kind of gesture. This can help the users intuitively understand how movements they are making are perceived by the device.

Using audio feedback it may be faster and it needs less material resources since there is no need to use any monitor, or external screen to visualize how user's gestures are being detected and interpretated. However, the audio feedback appeares to be that it is not as powerful and direct as the visual one.

In the present work, it's used two different sounds as audible feedback, one starting sound that is triggered when the user performs the *Wave*, and a second one that is emitted each time the user selects a button by performing one of the gestures *PointUp*, *PointMiddle* or *PointDown*.

Chapter 8

User study

In this section is described the user study that was followed in order to evaluate the system and the hypotheses reached in the present work. It is divided into three parts: The evaluation of the present work, its results and the comparison with the previous gesture control implementation using a Myo bracelet [45].

8.1. Evaluation process

The main goal of the user study is to evaluate the effectiveness and accuracy of the present work and to compare it with the Myo system. For doing so, both were evaluated with the same kind of questions regarding how comfortable, accurate and easy was to use each interactive system and which was the error rate and the latencies.

In the evaluation process, 15 testers took part in the experiment where each of them performed the test separately and individually. Firstly, they were explained which gestures could be used, how they would trigger the animations and how they had to perform them. Afterwards, they had to get familiarized with the usage of the system by repeating several times the sequence os gestures and finally, each of them had to stand next to a whiteboard where the virtual weather studio was projected and 2 meters separated from the Kinect sensor which was placed pointing directly to the actor. Then, the user had to perform the whole set of gestures while interacting with the virtual studio. First the *Wave* gesture which initialized the application and made appear the presentation display and the 3 buttons of the menu. Then, the used had to perform the *PointUp* gesture by pointing to the *Temperatur* button. By performing the *PointMiddle*, the video display was showed and the actor had to interact with the map using the *Scrolling* and *Zooming* set of gestures. Finnally, the last gesture to be tested was the *PointDown* which showed the *Morgen* display.

At the end of the test, it was submitted a questionnaire to each testing volunteer (see Appendix B) where they evaluated the interaction quality with the weather TV studio using the Kinect sensor. The questionnaire had two kinds of questions: closed answers (ranging from 1 to 5) where the users had to punctuate the answer with a discrete number; and open ones, where the people could develop and argument their answers.

8.2. Results

The use of the Kinect sensor by the volunteers was more difficult than expected. Although the sensor detected well all the volunteers, no matter how was their body shape, the synchronization of the gestures worked very differently. The pointing gestures were widely well recognized, but instead, the scrolling ones were more difficult to be recognized since each user performed them very differently and also were distant to the original gesture with what the system was trained as shown in Figure 8.1.

Which gesture was best detected?	
INTERACTION	**TOTAL SUM**
Starting	9
Pointing up	11
Pointing to the middle	10
Pointing down	10
Scrolling up	
Scrolling down	
Zooming in	5
Zooming out	3

Figure 8.1.: Number of volunteers that indicated witch was the gesture that was best detected.

By taking into account the open answers where the volunteers wrote their opinions and developed their interaction experiences (see Appendix B), it could be concluded that some subjects had thereby problems to learn well the gestures and perform them in such a way that the system could link the current gesture to the corresponding previously trained one. The 13% of the users detected a notable delay in the performance of the Scrolling continuous gestures, while the 25% detected it in the Zooming gestures. Both of these cases occurred in continuous gestures, instead, the discrete ones were more faster detected.

On another hand, the 30% of the volunteers felt that the tracking of the Scrolling gestures was not smooth and was clipped, this could be due to the way they performed such gesture.

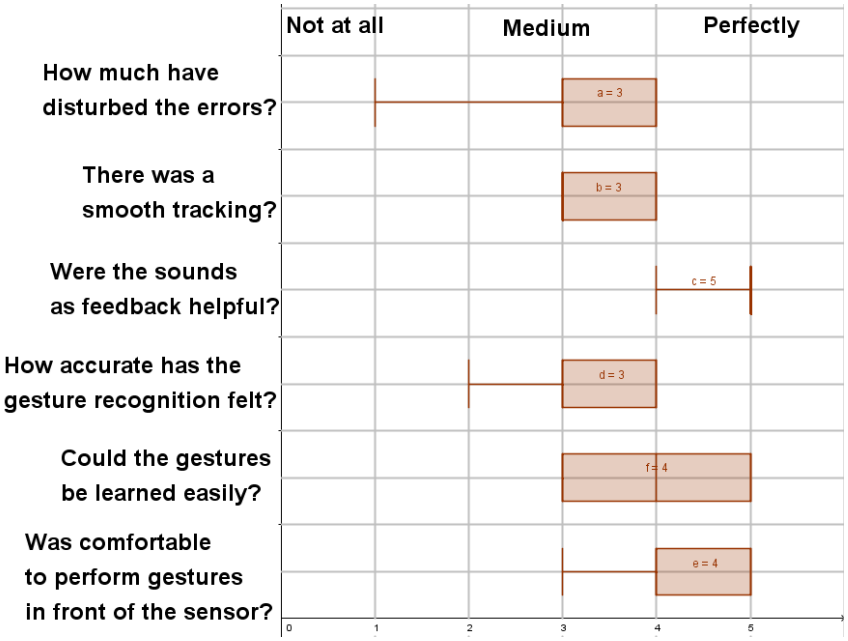

Figure 8.2.: Questions and evaluation of the Kinect sensor.

The boxplots shown in Figure 8.2 provide information about the rating with respect to the different categories of the gesture control using the Kinect sensor. It will be appreciated that the usage of the sensor is not perceived as disturbing. Moreover, the Kinect could not be considered as a disturbance factor, neither from a visual point of view, since the viewer would not be aware of which gesture recognition method is being used and this fact makes the Kinect be able to be used in real virtual TV studios (see Figure 8.3) without disturbing the spectators of the program.

In general, the gesture control using the Kinect sensor was a positive trend as can be seen in Figure

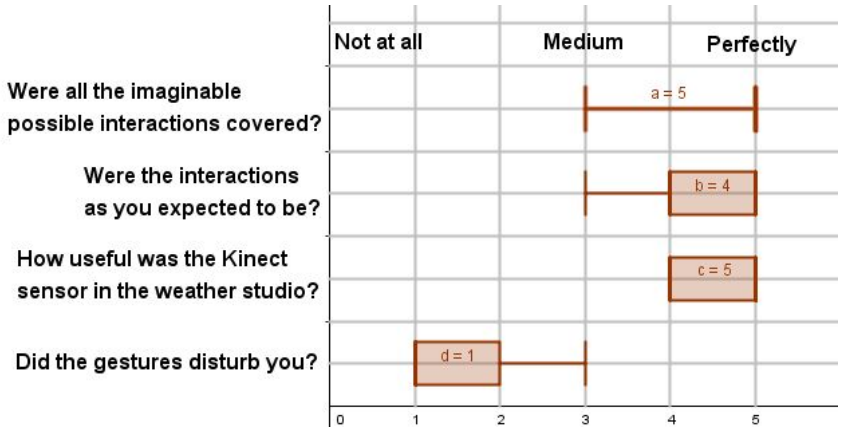

Figure 8.3.: Questions and evaluation of the Kinect sensor.

8.2. The majority of the volunteers felt the gesture recognition to be enough accurate and easy to learn. However, when using the sensor occurred repeatedly errors, these were due to the fact that the gestures were either incorrectly or not recognized. As seen in Figure 8.2 the repetitive mistakes were moderately annoying.

The provided feedback for the users was visual and acoustic and both of them were helpful and necessary for the interaction (see. Figure 8.2). However, it turned out that a visual feedback was more needed than the acoustic one to have a better orientation in the virtual space. However, the 100% of the subjects reported that no further feedback than the visual and acoustic ones was necessary.

Which interaction made more sense to you?	
INTERACTION	**TOTAL SUM**
Starting	8
Selecting between the buttons	12
Scrolling the map	7
Zooming the map	7

Figure 8.4.: Number of volunteers that indicated witch was the interaction that made most sense.

If the gestures are recognized, the volunteers reported that the tracking could easily be used. The interactions that were associated with changing between displays by selecting the buttons appeared to have more sense to the subjects (see Figure 8.4). There were no noticeable latency on them.

In summary, the Kinect sensor was received positively. There were some problems in the variety of clothing when detecting the skeleton and thereby the gestures as well as the distance and angle between the sensor and the subject could change the accuracy. The audio-visual feedback can be considered as necessary for the user in order to interact with the virtual studio.

8.3. Comparison

The Kinect sensor is then compared with the Myo bracelet in order to interact with the same virtual TV studio [12].

The Myo is a gesture control armband based on electromyography (EMG). It contains eight medical

grade stainless steel EMG sensor, a nine axis inertial measurement unit (IMU) which contains a three axis gyroscope, three axis accelerometer and a three axis magnetometer [1]. These units provide the orientation and movement of the wearer's arm. The orientation data indicates the positioning of the armband in terms of roll, pitch and yaw. With skin contact, the EMG sensors are able to read muscle activity via EMG data strings. Within the data strings the Myo armband is able to recognize special patterns which can be translated into gestures.

8.3.1. Gesture recognition with Myo

The five predefined standard gestures were trained with machine learning. These gestures were set by the developers, Thalmic Labs. They collected data from many different users trying the Myo armband, to build the algorithms they are using to recognize gestures. It was necessary to collect as much data as possible about every possible method of doing each of the selected gestures. With this method, the impact of factors that affect the EMG signal, like muscle anatomy, arm circumference, forearm strength or arm hair, can be reduced. Each user has an individual way to perform a gesture and the EMG patterns for the same gesture can vary widely between different users. Using machine learning, the range of the gesture recognition can be extended, so that every person is able to use the Myo armband.

With installing the Myo and setting it up for the first use, the user has to perform the set of gestures. This data string is collected and can be used to improve the performance of the gesture recognition. It is also possible to create a personal calibration profile, so that the five gestures are calibrated individually for a specific user.

In addition to the default gestures, it is possible to access the raw EMG data via the Myo SDK 0.9.0. In theory, with this and machine learning algorithms it is possible to create custom gestures that can be recognized even with different users. But in the contrary to the Kinect, described later in this paper, the Myo has no official software that is able to train and test new gestures at the moment [12].

8.3.2. Integration with Myo

To use the Myo within the virtual application, a communication between the software Viz Artist and the armband was necessary. The OSC protocol was used to accomplish that. With the MyOSC software, the armband is able to send and receive OSC messages. These messages contain information about the value of the orientation and the performed gestures. Viz Artist is also able to receive and process these messages.

This enables the usage of the gestures as a trigger for animations or events within the virtual application, as well as mapping virtual objects onto the orientation of the gesture armband.

Only two gestures were used to control the virtual application. *Fist* and *DoubleTap* in combination with the vertical orientation were enough to control the animations of the virtual weather studio. These two gestures were used to initialize a virtual event but, if these were combined with the orientation values, then were used to choose between the three buttons of the virtual set. The vibrotactile feedback of the Myo was used to indicate the user that a gesture was performed correctly.

[1] `https://www.thalmic.com/en/myo/`

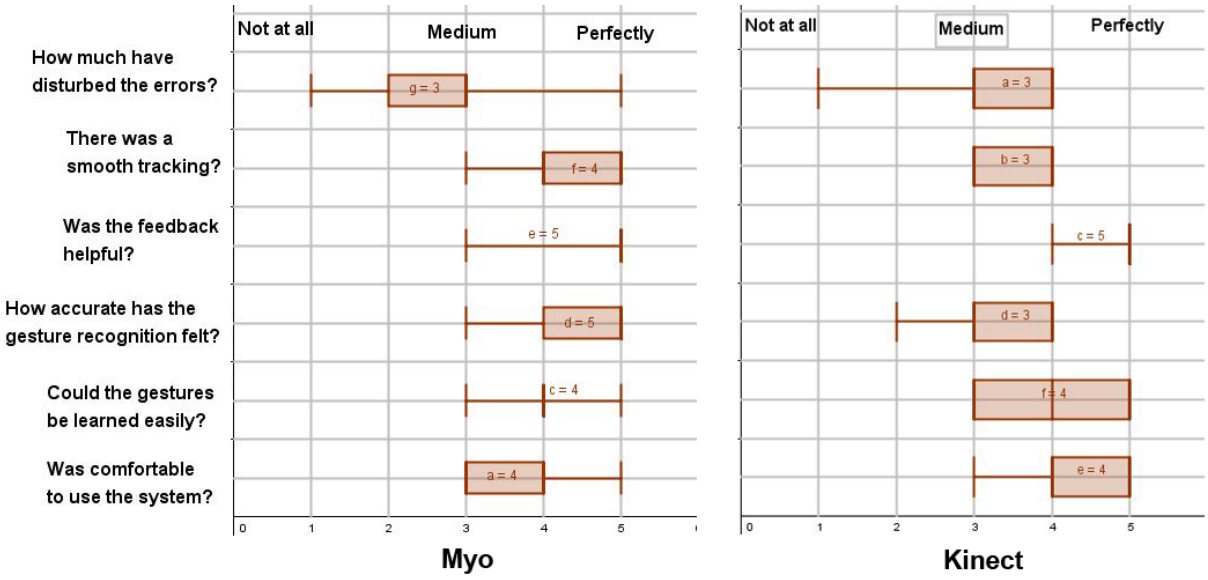

Figure 8.5.: Questions and evaluation of the Myo armband and the Kinect sensor

The results were as shown in Figure 8.5.

For comparing both technologies, it was made a *T-test*[2] assesses whether the means of two groups are statistically different from each other. This analysis is appropriate in order to compare the means of both groups. Since the two systems have been evaluated with the same virtual weather TV studio and the volunteers were subjected to similar questions, the means of both technologies can be directly compared. T tests, and related non-parametric tests can compare the two sets of measurements (where data is expressed using an interval or ratio scale). The results of applying a T-test to the means of the questions from the Table 8.1 are as follow: The T–test results in table 8.1 show that there is a significant

Question	$\bar{x}$ **Kinect**	$\bar{x}$ **Myo**	σ **Kinect**	σ **Myo**	**P value**
Was comfortable to use the system?	4.3	3.9	2	2	0.5882
Were the gestures easy to learn?	4.1	4.5	2	2	0.5882
How accurate has the gesture recognition felt?	3.5	2.7	1	2	0.1768
Was the feedback helpful?	4.9	4.7	1	1	0.5882
Was there a smooth tracking?	3.5	4.3	1	2	0.1768
How disturbing were the errors?	3.1	2.9	3	3	0.8564
How useful could be in a real TV studio?	4.7	4.2	1	2	0.3938
Did the gestures disturb you?	1.6	2.3	2	3	0.4584
How easily could the gestures sequence be learnt?	4.3	4.3	2	2	1

Table 8.1.: Values obtained for each question after evaluating both systems over 15 volunteers applying a T-test.

difference in the question referred to *"Was there a smooth tracking?"* and *"How accurate has the gesture recognition felt?"* ($\rho < 0.2$) and yet the other questions proof to be considered, by conventional criteria, not statistically significant ($\rho > 0.2$).

[2] `http://www.socialresearchmethods.net/kb/stat_t.php`

Here, a clear differences are visible and though, the need to perform a T–test for further evaluation of the results was present. How the test was performed is explained for the creation task below. The null–hypothesis is defined as:

$$H_0 : \mu_K = \mu_M$$

and means that the mean answers are equal for the entire sample volunteers for both Kinect (μ_M) and Myo (μ_M) systems. The alternative hypothesis, which directly correlates to the hypothesis of the present work, is defined as

$$H_A : \mu_K < \mu_M$$

and means that the fluency of the tracking is less when interacting using the Kinect, which in turn could indicate that the Myo interaction is more accurate. The alternative hypothesis defines this test as a one–tailed T–test since it predicts a direction [**?**]. To reject the null–hypothesis, the results of the user study have been used as a sample population. The sample means have the values $\bar{x}_K = 3.502$ and $\bar{x}_M = 4.355$.

The sample size of both groups is equal to $n_K = n_M = 15 = n$. The last thing needed for the T–test equation is the variance. The equation is defined as

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

where the sum with x_i values, the variances equal is cycling through all values for each participant. The variances for each technology are: $s_K^2 = 0.52$ and $s_M^2 = 0.72$. They are therefore almost equal, which makes the T–test of type 2: Different sets of participants for each group with equal variances. The equation for calculating the actual T–value is then defined as[3].

$$t = \frac{\bar{x}_K - \bar{x}_M}{\sqrt{\frac{(n-1)s_K^2+(n-1)s_M^2}{n+n-2} \cdot \frac{n+n}{n \cdot n}}}$$

Using this formula, the resulting T–value for the fluency of the tracking equals[4]:

t tracking= 3.77532

That makes the significance level (ρ):

ρ tracking = 0.00382

Which means that the difference between the means is highly significant ($\rho < 0.01$) which in turn causes the null-hypothesis ($\mu_K = \mu_M$) to be rejected and shows that the fluency of the tracking is highly significantly less with the Kinect system than with the Myo bracelet.

Overall, the tested people had a broadly positive experience. The arm tracking using the Myo was accurate and the vibrations were helpful to give the user a feedback information about what was happening in the virtual environment. The precision of the gesture recognition was rated positively and the gestures were easy to learn. Some difficulties have been encountered while initializing the Myo armband because of the varying muscle anatomy of the test people. After testing and using the Myo for some time the gesture recognition for every test person could be improved [45].

[3] http://www.ruf.rice.edu/~bioslabs/tools/stats/ttest.html
[4] http://www.socscistatistics.com/tests/studentttest/

The Kinect system was also qualified as comfortable, easy and intuitive to use, however, such a visual-based gesture recognition system is sensible to the variation of heights and clothes. Also could be appreciated that the distance and angle between the user and the background played an important role in the detection of the gesture.

Overall, every tester was able to control the virtual environment with both systems. However, for a live TV broadcasting production there are still too many possible errors, like false gesture recognition, which can disrupt the experience. For reducing the impact of these errors, further calibration and testing is needed with the Myo, as well as a wider training recording much variety of people using the Kinect sensor.

Chapter 9

Adversities

In this section are exposed the problems that were presented through the implementation of the interactive system using Machine Learning.

One of the main problems was the hand state accuracy of the Kinect sensor. Kinect can differentiate between open, closed and pointing hand state but this last one did not have a very precise accuracy [46] which made the implementation of gestures such as *PointUp*, *PointMiddle* and *PointDown* difficult to train since not always detected correctly the corresponding hand state. The quality of pointing calculations heavily depended on the viewpoint of the sensor. This problem was solved by combining the simple action of pointing with the right hand, jointly with an open left hand. In such a way, combining the state of both hands, the system could recognize and differentiate such gesture with a higher accuracy.

A notable mistake that was disturbing in the Kinect framework using Machine Learning algorithms to recognize gestures, was the fact that the user had to perform the gestures exactly, or at least very similar, to the ones that were used to train the system. If they were not very alike, the system had problems to recognize the gestures and thereby, the interaction was not smooth. This had to be sorted out by making the user spend some time training and getting familiarized with the sequence of gestures until they were well performed.

Another issue that had to be taken into account was the fact that the last segment of the *ScrollUp* and *ScrollDown* gestures was initially confused with the final segment of the *ZoomIn* gesture. The solution was to also relapse into the hand state, this means that the *ZoomIn* and *ZoomOut* had to be trained with a closed hand and *ScrollUp* and *ScrollDown* with a different one, in this case, it was chosen an open state.

The last noteworthy problem was that the system did not recognize in the same way a certain gesture performed by different people with varied physiology, shape and colour. This made that the gestured had to be recorded with several people, with a variety of clothes and different colours. Also, the Kinect has its own limitation. The body joints defined by Kinect are not precise when compared with a marker-based system [47]. As Kinect v2 works on the principle of depth range using time-of-flight technology and detects human presence by recognizing the body posture, in some cases it might create skeleton on some non-human objects like tables or chair especially when object parts might look like extremities. This was solved by locating the actor sufficiently far from the background and other objects.

Chapter 10

Conclusion

In the present work an approach for a gesture recognition navigation in a virtual reality TV studio has been described and compared with a previous work made by a different gesture recognition sensor, the Myo bracelet. Both use machine learning algorithms and have been compared in the same virtual TV studio application as well as the effectiveness and experience of controlling the virtual set was evaluated by performing a user test.

The two kinds of installations, the Kinect sensor and the Myo bracelet, are inexpensive solutions. An advantage of the Myo is that the interaction space is not limited to a small tracking area and, as the Kinect is completely vision-based, the user does not have to learn how to use a new device, just has to pass through a short phase of getting familiarized with the intuitive gestures.

Interacting with the system using the Kinect sensor allowed the user to perform more intuitive gestures than with the Myo armband, however, the tracking was less smooth due to the wide variety of shapes and sizes a user can have, instead, the Myo bracelet could adapt better to this variation. Using the Microsoft Kinect and the Myo armband, a method for interaction control using the user's gestures and postures is designed and implemented.

The use of machine learning for gesture recognition opens up a new paradigm even for complicated signals like EMG or complex movements which can be used easily as input. Feedback has proven to support the interaction with the interactive system. For the Kinect sensor it is necessary to use either visual or auditory feedback but in any cases, it has been demonstrated that feedback is necessary to make the users aware of the correct detection of the gestures.

Also, it has been made a comparison between heuristic and machine learning methods for gesture recognition.

The process of recording gesture profiles includes defining a starting point, transition points along the way and an end point by recording skeletal joint positioning in the key frames. Timing from start to end as well as point to point is measured along with data points that describe the skeleton joint positions (or pose) demarking the transition states. For example, a very simple gesture has four transition points; a start, an end and two intermediate states.

From a machine learning point of view, if an actor successfully moves from the starting point through the two intermediate states and finally the ending state, in the described time periods for each segment and the overall gesture, the gesture will be recognized algorithmically. The problem with this approach for gesture detection is that the recorded gesture would be specific to the actor who recorded it and only those with a similar frame would successfully have the gesture detected. Meaning each required gesture would need to be configured for each person, resulting in a gesture profile for that actor.

On another hand, the advantages of heuristics methods should be obvious: they tend to be simple to explain, understand and simple to code. The problem, of course, is that without a theoretical justification, it can be hard to tell if an heuristic system is really as good as it is wanted to be, so the correct answer to a

performed gesture is often far different from the value suggested by the standard heuristic.

In a number of cases, it turns out that it is not possible to come up with an algorithm that is simple enough to define, with an acceptable confidence and high accuracy, a complex gesture in any situation. Instead, it will just manage to closely approximate the value. Thus, performing an heuristic method for gesture recognition applications is such a costly, elaborated and low-efficient work.

However, machine learning, as the name suggest, needs minimal human effort and makes this process much easier, faster and with more accurate results. It is enough simply to record the same gesture several times to let the algorithm "learn", drawing on the experience, what is the gesture and what is not. Machine learning works on iterations where computer tries to find out patterns hidden in data. Because of the machine does this work on comprehensive data and is independent of all the assumption, predictive power is generally very strong for these models. Instead, heuristics models are mathematical intensive and have a strong theoretical and descriptive basement. It requires the modeller to understand the relation between variables before putting it in.

Machine learning techniques are particularly useful in gesture recognition applications since a whole gesture is too complex to be described by analytical formulations or manual brute force design, and is dependent to the environment in which it is deployed. Gesture recognition is a suitable environment to “act by learning” rather than being explicitly programmed. As it has been described, machine learning offers palpable advantages compared to facing a gesture from an heuristic point of view directly mapping the gestures provided by complex data generated by sensor systems and interactive interfaces.

Conslusión:

En el presente trabajo se ha implementado un sistema de reconocimiento de gestos para interactuar con un estudio de televisión de realidad virtual y se ha comparado con un trabajo previo realizado por un sensor de reconocimiento de gestos diferente, el brazalete Myo. Ambos hacen uso de algoritmos de aprendizaje de máquinas y son comparados en una aplicación de estudio de TV virtual, así como también ha sido evaluada la eficacia y la experiencia de controlar el decorado virtual.

La instalación del sensor Kinect y el Myo es una solución de bajo costo. Una ventaja del uso del brazalete Myo es que el espacio de interacción no se limita a una pequeña área, no obstante, como el sensor Kinect está completamente basado en la reconocimiento de gestos visual, el usuario no tiene que aprender a utilizar un nuevo dispositivo, sólo ha de pasar por un corto periodo de familiarización intuitiva con los gestos.

Interactuar con el sistema usando el sensor de Kinect permite al usuario realizar gestos más intuitivos que con el brazalete Myo, sin embargo, el seguimiento fue menos preciso debido a la gran variedad de formas y tamaños que un usuario puede tener, en cambio, la pulsera Myo pudo adaptarse mejor a esta variación. Utilizando el sensor Kinect de Microsoft y el brazalete Myo, se ha diseñado e implementado un método para el control de la interacción mediante gestos y posturas del usuario.

El uso de la máquina de aprendizaje para el reconocimiento de gestos abre un nuevo paradigma incluso

para señales complicadas como señales electromiográficas (EMG) o movimientos complejos que se pueden utilizar fácilmente como entrada. La evaluación ha demostrado que el uso de feedback vibrotáctil en el sistema Myo es necesario para apoyar la interacción con el sistema interactivo. Para el sensor de Kinect es necesario usar la retroalimentación visual o auditiva, pero en cualquier caso se ha demostrado que el uso de feedback es necesario para hacer que los usuarios sean conscientes de la correcta detección de los gestos.

También se ha hecho una comparación entre métodos heurísticos y de aprendizaje de máquinas para el reconocimiento de gestos.

El proceso de grabación de perfiles gestuales incluye la definición de un punto de inicio, varios puntos de transición a lo largo del gesto y un punto final mediante el registro de las posiciones de las articulaciones del esqueleto en los diversos fotogramas. Las marcas de tiempo desde el punto inicial al final así como de punto a punto son medidas con puntos de datos que describen la posición de la articulación del esqueleto (o "postura") delimitando los estados de transición. Por ejemplo, un gesto muy simple tiene cuatro puntos de transición; un inicio, un final y dos estados intermedios.

Desde el punto de vista del aprendizaje de máquinas, si un actor se mueve con éxito desde el punto de partida, a través de los dos estados intermedios, al estado final en los períodos de tiempo descritos para cada segmento y el gesto general, el gesto se reconocerá afirmativamente de manera algorítmica. El problema con este enfoque para la detección de posturas y gestos es que el gesto registrado sería específico para el actor que lo grabó, y los gestos únicamente de aquellos usuarios con una forma y tamaño similar, podrían ser detectados con éxito. Es decir, cada gesto requerido tendría que ser configurado para cada persona, lo que resulta en un perfil de gestos especial para cada actor.

Las ventajas de los métodos heurísticos pueden parecer obvias: tienden a ser más fácil de explicar, entender y codificar. El problema, por supuesto, es que sin una justificación teórica, puede ser difícil saber si un sistema heurístico es realmente tan bueno como se requiere. No obstante, la respuesta correcta a un gesto realizado es a menudo muy diferente del valor sugerido por la heurística estándar.

En varios casos, resulta que no es posible llegar a un algoritmo que es lo suficientemente simple como para definir, con un porcentaje de confianza y precisión aceptables, un gesto complejo en cualquier situación. En cambio, únicamente podrá aproximar el valor correcto. Por lo tanto, emplear un método heurístico en aplicaciones dónde se requiera de reconocimiento de gestos es un trabajo costoso, elaborado e ineficiente.

Sin embargo, el aprendizaje de máquinas, como el nombre sugiere, necesita un mínimo esfuerzo humano y hace que este proceso sea mucho más fácil, más rápido y con resultados más precisos. Es suficiente con registrar el mismo gesto varias veces para que el algoritmo "aprenda", basándose en la experiencia, lo que es el gesto y lo que no lo es. El aprendizaje automático funciona en iteraciones, donde el sistema intenta descubrir patrones ocultos en los datos. Debido a que el sistema realiza dicho trabajo de manera exhaustiva e independientemente de toda suposición, generalmente la predicción toma un importante papel en este tipo de métodos. En cambio, los modelos heurísticos tienen una gran parte de matemática y una fuerte base teórica y descriptiva. Requieren de una persona humana que realice un estudio de la relación entre las variables antes de construir el algoritmo.

Es por ello por lo que las técnicas de aprendizaje de máquinas son particularmente útiles en aplicaciones que requieran de reconocimiento de gestos ya que describir todo un gesto mediante formulaciones analíticas es demasiado complejo y depende del entorno en el cual se registra. El reconocimiento de gestos es un entorno adecuado para actuar mediante el aprendizaje en lugar de ser programado de forma explícita.

Como ya se ha descrito, el aprendizaje de máquinas ofrece considerables ventajas en comparación a afrontar un gesto desde un punto de vista heurístico mapeando directamente los gestos proporcionados por los datos complejos generados en un sistema de sensores e interfaces interactivas.

Conslusió:

En el present treball s'ha implementat un sistema de reconeixement de gestos per interactuar amb un estudi de televisió de realitat virtual i s'ha comparat amb un treball previ realitzat amb un sensor de reconeixement de gestos diferent, el braçalet Myo. Tots dos fan ús d'algoritmes d'aprenentatge de màquines i són comparats en una aplicació d'estudi de TV virtual, així com també ha estat avaluada l'eficàcia i l'experiència de controlar el decorat virtual.

La instal·lació del sensor Kinect i el Myo és una solució de baix cost. Un avantatge de l'ús del braçalet Myo és que l'espai d'interacció no es limita a una petita àrea, però, com el sensor Kinect està completament basat en la reconeixement de gestos visual, l'usuari no ha d'aprendre a utilitzar un nou dispositiu, només ha de passar per un curt període de familiarització intuïtiva amb els gestos.

Interactuar amb el sistema utilitzant el sensor de Kinect permet a l'usuari realitzar gestos més intuïtius que amb el braçalet Myo, però, el seguiment va ser menys precís a causa de la gran varietat de formes i mides que un usuari pot tenir, en canvi, la polsera Myo va poder adaptar-se millor a aquesta variació. Utilitzant el sensor Kinect de Microsoft i el braçalet Myo, s'ha dissenyat i implementat un mètode per al control de la interacció mitjançant gestos i postures de l'usuari.

L'ús de la màquina d'aprenentatge per al reconeixement de gestos obre un nou paradigma fins i tot per a senyals complicades com senyals electromiogràfiques (EMG) o moviments complexos que es poden utilitzar fàcilment com a entrada. L'avaluació ha demostrat que l'ús de feedback vibrotàctil en el sistema Myo és necessari per donar suport a la interacció amb el sistema interactiu. Per al sensor de Kinect cal fer servir la retroalimentació visual o auditiva, però en qualsevol cas s'ha demostrat que l'ús de feedback és necessari per fer que els usuaris siguin conscients de la correcta detecció dels gestos.

També s'ha fet una comparació entre mètodes heurístics i d'aprenentatge de màquines per al reconeixement de gestos.

El procés de gravació de perfils gestuals inclou la definició d'un punt d'inici, diversos punts de transició al llarg del gest i un punt final mitjançant el registre de les posicions de les articulacions de l'esquelet en els diversos fotogrames. Les marques de temps des del punt inicial al final així com de punt a punt són mesures amb punts de dades que descriuen la posició de l'articulació de l'esquelet (o "postura") delimitant els estats de transició. Per exemple, un gest molt simple té quatre punts de transició; un inici, un final i dos estats intermedis.

Des d'un punt de vista heurístic, si un actor es mou amb èxit des del punt de partida, a través dels dos estats intermedis, a l'estat final en els períodes de temps descrits per a cada segment i el gest general, el gest es reconeixerà afirmativament de manera algorísmica. El problema amb aquest enfocament per a la detecció de postures i gestos és que el gest registrat seria específic per a l'actor que el va gravar, i els

gestos únicament d'aquells usuaris amb una forma i mida similar, podrien ser detectats amb èxit. És a dir, cada gest requerit hauria de ser configurat per a cada persona, el que resulta en un perfil de gestos especial per a cada actor.

Els avantatges dels mètodes heurístics poden semblar obvis: tendeixen a ser més fàcils d'explicar, entendre i codificar. El problema, per descomptat, és que sense una justificació teòrica, pot ser difícil saber si un sistema heurístic és realment tan bo com es requereix. No obstant això, la resposta correcta a un gest realitzat és sovint molt diferent del valor suggerit per l'heurística estàndard.

En diversos casos, resulta que no és possible arribar a un algoritme que siga prou simple com per definir, amb un percentatge de confiança i precisió acceptables, un gest complex en qualsevol situació. En canvi, únicament podrà aproximar el valor correcte. Per tant, emprar un mètode heurístic en aplicacions on es requereixi de reconeixement de gestos és un treball costós, elaborat i ineficient.

No obstant això, l'aprenentatge de màquines, com el seu propi nom suggereix, necessita un mínim esforç humà i fa que aquest procés sigui molt més fàcil, més ràpid i amb resultats més precisos. Hi ha prou amb registrar el mateix gest diverses vegades perquè l'algoritme "aprenga", basant-se en l'experiència, el que és el gest i el que no ho és. L'aprenentatge automàtic funciona en iteracions, on el sistema intenta descobrir patrons ocults en les dades. A causa que el sistema realitza aquesta feina de manera exhaustiva i independentment de tota suposició, generalment la predicció pren un important paper en aquest tipus de mètodes. En canvi, els models heurístics tenen una gran part de matemàtica i una forta base teòrica i descriptiva. Requereixen d'una persona humana que realitze un estudi de la relació entre les variables abans de construir l'algoritme.

És per això pel que les tècniques d'aprenentatge de màquines són particularment útils en aplicacions que requereixen de reconeixement de gestos ja que descriure tot un gest mitjançant formulacions analítiques és massa complex i depèn de l'entorn en el qual es registra. El reconeixement de gestos és un entorn adequat per a actuar mitjançant l'aprenentatge en lloc de ser programat de manera explícita. Com ja s'ha descrit, l'aprenentatge de màquines ofereix considerables avantatges en comparació a afrontar un gest des d'un punt de vista heurístic mapejant directament els gestos proporcionats per les dades complexes generats en un sistema de sensors i interfícies interactives.

Appendix A

Machine learning algorithms

In this section it will be described the use of three of the most common Machine Learning (ML) methods used to recognize postures and gestures. ML algorithms tries to improve their performance based on experience [48]. This field of study finds an application in problems where existing models are deficient and it's not possible to implement specialized algorithms, and where, however, there is plenty of real data. It is common to find tasks that are easy for humans, but are difficult to explain and therefore have no efficient algorithmic solution.

The algorithms based on supervised learning, are designed to construct a function $f : X \to Y$ from a set of known inputs and outputs (x, y) called *training set*. When f is a continuous function, it's all about a regression problem; however, if f is a discrete function, is a classification problem.

In a classification problem, elements from Y set are interpreted as classes or categories to which belong the elements of the set X. therefore, the f function is called a *classifier*. Gesture recognition can be modelled as a classification problem where the training set are formed by pairs (x, y), where x is a gesture (represented by a set of numeric features) and y is the gesture's class.

A.1. Neural Networks

This section presents a brief introduction into the concepts involved in neural networks which are among the most important ML techniques. These are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. These connections are not all equal: each connection may have a different strength or *weight*. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called *nodes*; the nodes are connected by links, and the links have an associated weight that can act as a storage mechanism [49]. Each node is considered a single computational unit containing two components. The first component is the input function which computes the weighted sum of its input values; the second is the activation function, which transforms the weighted sum into a final output value.

Commonly neural networks are trained so a that a particular input leads to a specific target output. Such a situation is illustrated in Figure A.1. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are used, in this *supervised learning*, to train a network.

A.1.1. Neural Network Topologies

Neural networks generally have two basic structures or topologies, a feed-forward structure and a recurrent structure. A feed-forward network (see Figure A.2) can be considered a directed acyclic graph,

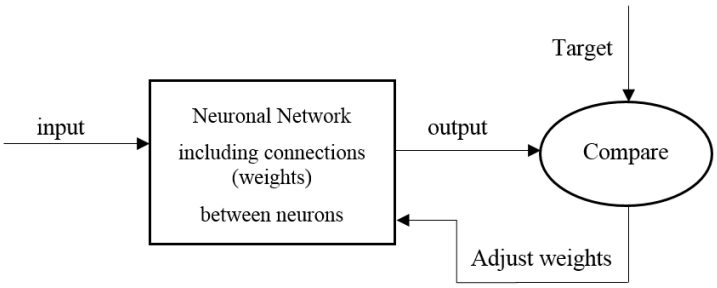

Figure A.1.: Neural Network block diagram

while a recurrent network has an arbitrary topology. The first one consists of units, organized in *layers* and every unit in a layer is connected with all the units in the previous layer. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called *feed-forward* neural networks.

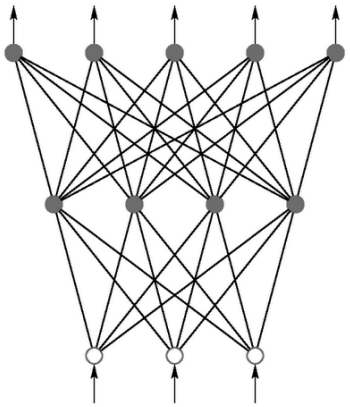

Figure A.2.: 2-layered network with an output layer with 5 units, a hidden layer with 4 units and 3 input units. [13]

The recurrent network has the advantage over a feed-forward network in that it can model systems with state transitions. However, recurrent networks require more complex mathematical descriptions and can exhibit chaotic behaviour. In both network topologies, there is no restriction on the number of layers in the network. These multi-layered networks provide more representation power at the cost of more complex training. The nodes in between the input and output nodes of the multi-layered network have no communication with the outside world and cannot be directly observed from the input or output behaviour of the network. If the number of hidden nodes is large, it is possible to represent any continuous function of the inputs [49].

A.1.2. Learning process

- Supervised learning:

  The feed-forward network uses a supervised learning algorithm: besides the input pattern, the neural network also needs to know to what category the pattern belongs. Learning proceeds as follows: A pattern is presented at the inputs. The pattern will be transformed in its passage through the layers

of the network until it reaches the output layer. All this units of the output layer belong to a different category and they are now compared with the outputs as they ideally would have been if this pattern would be correctly classified: in this case the unit with the correct category will have the highest output value and the output values of the other output units will have very small ones. On the basis of this comparison, all the connection weights are modified a little bit to guarantee that, the next time this same pattern is presented at the inputs, the value of the output unit that corresponds with the correct category is a little bit higher than it's now and that, at the same time, the output values of all the other incorrect outputs are a little bit lower than they are now. The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers to be used at these layers to modify connection weights. This is why the term *back-propagation network* is also often used to describe this type of neural network.

If it is performed the procedure above once for every pattern and category pair in the data set that has been performed one epoch of learning.

Eventually, after many epochs, the neural network will come to remember these pattern-category pairs, and when the learning phase has finished, will be able to *generalize* and has learned to classify correctly any unknown pattern presented to it. Because real-life data often contains noise as well as partly contradictory information, this ideal situation can be fulfilled only partly.

- Unsupervised learning:

  This kind of learning mechanism trains the network to respond to a clusters of patterns within the input. There is no training in advance and the system must develop its own representation of the input, since no matching output is provided [50].

Generally, it's hard to know how long the learning phase will take place because it depends on the size of the neural network, the number of patterns to be learned, the number of epochs, the tolerance of the minimizer and the speed of the computer.

However, supervised and unsupervised learning do not have to be mutually exclusive: depending on the network, a combination of the two learning strategies can be employed.

A.1.3. Classification process

In the classification phase, the weights of the network are fixed and in contrast to the learning process, classification is very fast. A pattern, presented at the inputs, will be transformed from layer to layer until it reaches the output layer. Now classification can occur by selecting the category associated with the output unit that has the largest output value.

Neural networks are a useful method for recognizing hand and body postures and gestures, yield increased accuracy conditioned upon network training. However, they have distinct disadvantages. First, different configuration of a given network can give very different results, and it is difficult to determine which configuration is best without implementing them. Another disadvantage is the considerable time involved in training the network.

Finally, the whole network must be retrained in order to incorporate a new posture or gesture. If the

posture or gesture set is known beforehand, this is not an issue, but if postures and gestures are likely to change dynamically as the system develops, a neural network is probably not appropriate.

- Strengths

    1. Uses information about how humans interact with the physical world to help identify gestures.

- Weaknesses

    1. Uses only a limited set of gestures.

    2. Does not use hand orientation and position data or finger data.

    3. Does not run in real time.

A.1.4. Support vector machines

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in Table **??**. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling, for example, to the right is labelled as GREEN.

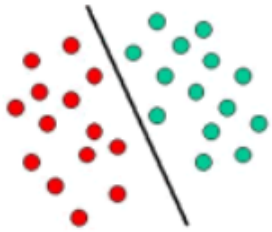

Table A.1.: Linear classifier. [14]

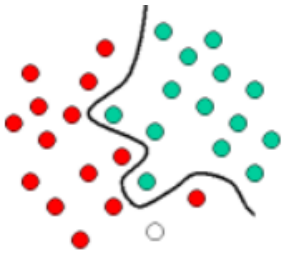

Table A.2.: Non-linear classifier. [14]

This is a classic example of a linear classifier, that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Is in this way, the Support Vector Machine technique is a linear classifier that maximizes the margin (see Figure A.3) and try to find two planes that separate correctly the training set without any spot between them. As much distance between the margins, more stable will be the system against input perturbations.

However, most of the classification tasks are not that simple and often more complex structures are needed in order to make an optimal separation, and correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in Table A.2. Compared to the

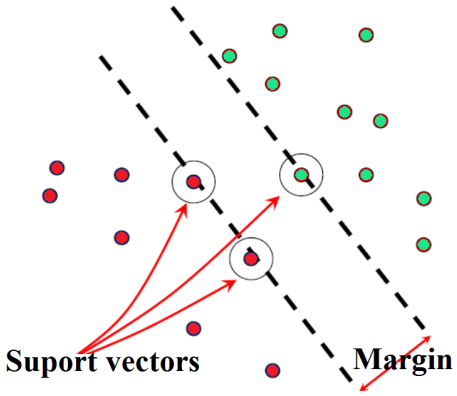

Figure A.3.: Linear classifier that maximizes the margin between borders.

previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve. Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

The basic idea behind Support Vector Machines is showed in Figure A.4.3. Here it can be appreciated the original objects (left side of the schematic) mapped, rearranged using a set of mathematical functions, known as *kernels*. The process of rearranging the objects is known as *mapping* (transformation). In this new setting, the mapped objects (right side) is linearly separable and, thus, instead of constructing the complex curve (left side), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

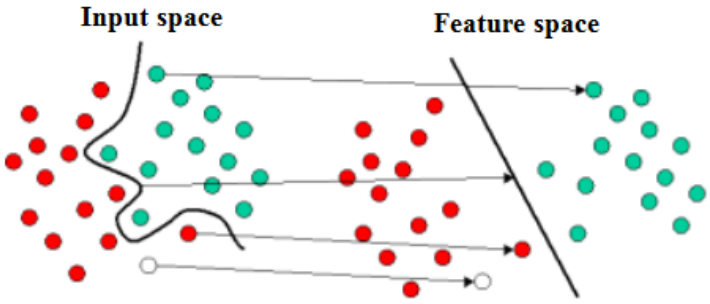

Figure A.4.: Linear classifier that maximizes the margin between borders.[14]

Support Vector Machine is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. It can be defined a non-linear model by transforming the data to a new plane with a larger amount of dimensions. Support Vector Machine supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a model variable is created with case values as either 0 or 1. Thus, a categorical dependent variable consisting of three levels, say (A, B, C), is represented by a set of three model variables:

$$A : \{1\ \ 0\ \ 0\},\quad B : \{0\ \ 1\ \ 0\}, C : \{0\ \ 0\ \ 1\}$$

To construct an optimal hyperplane, Support Vector Machine employs an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, Support Vector Machine models can be classified into four distinct groups [14].

- Classification SVM Type 1 (C-SVM classification)
- Classification SVM Type 2 (nu-SVM classification)
- Regression SVM Type 1 (epsilon-SVM regression)
- Regression SVM Type 2 (nu-SVM regression)
- Strengths
    1. Effective in high dimensional spaces.
    2. Still effective in cases where number of dimensions is greater than the number of samples.
    3. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
    4. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.
- Weaknesses
    1. If the number of features is much greater than the number of samples, the method is likely to give poor performances.
    2. Costly training process in non-linear Support Vector Machine systems.

A.1.5. Naïve Bayes Classifier

The Naïve Bayes Classifier uses the Bayes theorem in order to deduce the probability that an object belongs to a class or not; the classification rule consists in assigning to each object the class which the probability is higher. This idea will be explained with more details afterwards. The NBC is particularly suited when the dimensionality of the inputs is high. According to the Bayes theorem and considering the example displayed in Figure A.5, the objects can be classified as either GREEN or RED. The main task is to classify new cases as they arrive and decide to which class label they belong based on the currently existing objects.

Since there are twice as many GREEN objects as RED, it is reasonable to think that a new case (which has not been observed yet) is twice as likely to belong to the GREEN class rather than the RED one. In Bayesian analysis, this belief is known as the *prior probability*. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen. Thus, considering the previous example, can be expressed as:

$$Prior\ probability\ for\ GREEN \propto \frac{Number of\ GREEN\ objects}{Total\ number\ of\ objects}$$

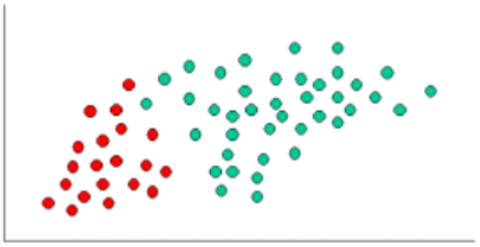

Figure A.5.: Example of a GREEN and RED objects to be classified by a Naïve Bayes Classification method. [14]

$$Prior\ probability\ for\ RED \propto \frac{Number of\ RED\ objects}{Total\ number\ of\ objects}$$

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, the prior probabilities to class membership are:

$$Prior\ probability\ for\ RED \propto \frac{40}{60}$$

$$Prior\ probability\ for\ RED \propto \frac{20}{60}$$

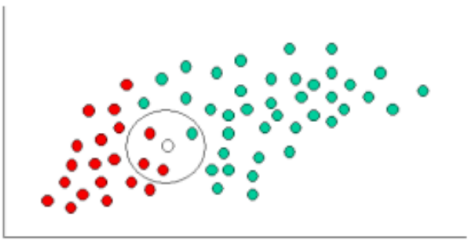

Figure A.6.: Classification of a new object (WHITE circle) [14].

Once it has been formulated the probability, the system is ready to classify a new object. Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular class. To measure this likelihood, there are considered a certain number (to be chosen a priori) of points irrespective of their class labels that have been indicated in this case as the ones inside the circle in Figure A.6. Then, it's calculated the number of points in the circle belonging to each class label. From this it's calculated the likelihood:

$$Likelihood\ of\ X\ given\ GREEN \propto \frac{Number\ of\ GREEN\ in\ the\ vicinity\ of\ X}{Total\ number\ of\ GREEN\ cases}$$

From the first equation it can be seen that $Likelihood\ of\ X\ given\ GREEN$ is smaller than $Likelihood\ of\ X\ given\ RED$, since the circle encompasses 1 GREEN object and 3 RED ones. Thus can be expressed as:

$$Probability\ of\ X\ given\ GREEN \propto \frac{1}{40}$$
$$Probability\ of\ X\ given\ RED \propto \frac{3}{20}$$

Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classificatory is produced by combining both sources of information, the prior and the likelihood, to form a posterior probability using the Bayesian theorem.

$$Posterior\ probability\ of\ X\ being\ GREEN \propto$$
$$Prior\ probability\ of\ GREEN\ x\ Likelihood\ of\ X\ given\ GREEN = \frac{4}{6}\frac{1}{40} = \frac{1}{60}$$

$$Posterior\ probability\ of\ X\ being\ RED \propto$$
$$Prior\ probability\ of\ GREEN\ x\ Likelihood\ of\ X\ given\ RED = \frac{2}{6}\frac{3}{20} = \frac{1}{20}$$

Finally, X is classified as RED since its class membership achieves the largest posterior probability.

- Strengths

  1. Very simple and fast system. If the Naïve Bayes conditional independence assumption actually holds, a Naïve Bayes classifier will converge quicker than discriminative models like logistic regression.

  2. There is needed less training data than the rest of methods.

  3. Not sensitive to irrelevant features.

- Weaknesses

  1. It can't learn interactions between features.

A.2. Boosting

Boosting means to combine weak classifiers (simple) in order to create a stronger classifier (more complex). A weak learner is defined to be a classifier which is only slightly correlated with the true classification, this means that has a performance value of only slightly above 50% of confidence and uses a simple threshold function. In contrast, a strong learner is a classifier that is arbitrarily well-correlated with the true classification. While boosting is not algorithmically constrained, most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners' accuracy. After a weak learner is added, the data is reweighted: Misclassified features are given bigger weights and correctly classified features are given lower weights for the next training round.

The method results in a final classifier that combines in an optimal way the strengths of the individual classifiers.

A.2.1. AdaBoost Algorithm

AdaBoost (Adaptive Boosting) is a Machine Learning algorithm to build a classifier based on boosting, a technique which was introduced by Freund and Schapire in 1999 [51]. An example of the stages in boosting are showed in Figure A.7 where after each classifier, the weight of each training data incorrectly classified is increased. The final classifier is a linear weighted combination of the weak classifiers:

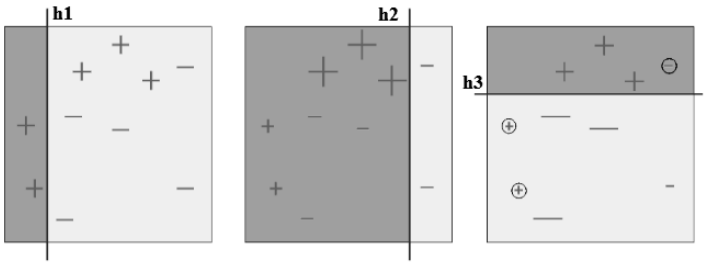

Figure A.7.: Example of three different weak classifiers.[15]

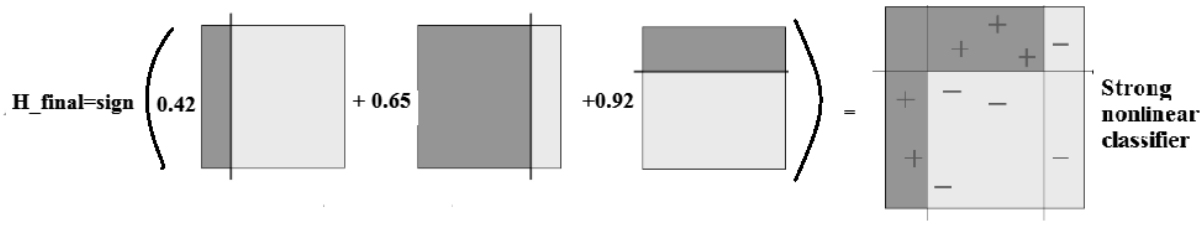

Figure A.8.: Illustration of stages in boosting. [15]

AdaBoost algorithm fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learners until a limit is reached in the number of models or accuracy.

The algorithm for a discrete version of AdaBoost runs as below:

1. Input of the training examples with labels $\{(x_1, y_1), (x_2, y_2), ...(x_i, y_i)\}$, where x_i is a feature vector in the feature space X and y_i is a label from Y={-1 or +°} for negative or positive results.

2. Initialization of all the weights $D_1(i) = 1/(n)$, where n is the number of elements in the feature vector.

3. For $t = 1, 2, ...T$ (T is the number of rounds):

    a) Training a weak classifier h_t, using the weights in D_1 so that $h_t \in \{-1, +1\}$.

b) Computation of the error associated with the weak classifier (the sum of the elements' weights that are incorrectly classified):

$$error_t = \sum_{i=0}^{n} D_t(i) h_t(x_i) y_i$$

c) Computation of the weight factor α_t for the classifier:

$$\alpha_t = 0.5 ln(\frac{1 - error_t}{error_t})$$

d) Update the weights $D_{t+1}(i)$ such that D_{t+1} is a new distribution, and the weights decrease if the element is correctly classified and increase if is incorrectly classified:

$$D_{t+1}(i) = D_t(i)^{-\alpha_t y_t h_t(_i)}$$

e) Re-normalize the weights:

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{\sum_{i=0}^{n} D_{t+1}(i)}$$

4. After T rounds, output the final classifier as a linear combination of selected classifiers:

$$H(x) = sign[\sum_{i=0}^{n} \alpha_t h_t(x)]$$

. In this way it combines the outputs from weak learner and creates a strong learner which eventually improves the prediction power of the model. Boosting pays higher focus on examples which are mis-classfied or have higher errors by preceding weak rules.

In order to improve some defect classifications using machine learning classifiers, AdaBoost algorithms can perform better compared to support vector machines and Bayesian classifiers, taking less time with more accuracy [52]. It was also recently combined with scale invariant feature transform features to improve multi-class hand and body posture recognition for human-robot interaction [53].

- Strengths [54]

1. Can achieve similar classification results with much less tweaking of parameters or settings than other Machine Learning methods.

2. The user only needs to choose:

   a) Which weak classifier might work best to solve their given classification problem.

   b) The number of boosting rounds that should be used during the training phase.

3. Less susceptible to the overfitting problem than most learning algorithms.

4. Fairly good generalization.

- Weaknesses

    1. Can be sensitive to noisy data and outliers.

    2. Does not support null rejection.

Appendix B

Algorithmic Techniques

Once the raw data has been collected from a vision-based system, it must be analysed to determine if any postures or gestures have been recognized. In this section, various algorithmic techniques for recognizing hand postures and gestures are discussed. The techniques in this survey fall into three rough categories:

1. Feature extraction, statistics and models

2. Learning algorithms

3. Miscellaneous techniques

B.1. Feature Extraction, Statistics and Models

This category contains six of the most common techniques for posture and gesture recognition that extract some mathematical quantity from the raw data. In these cases, the mathematical quantity is represented as a feature, a statistic, or a model.

B.1.1. Temporal Template Matching

The simplest method is through Template Matching [55] which consists on a high-level machine vision technique that identifies the parts on an image that match a predefined template. Advanced template matching algorithms allow to find occurrences of the template regardless of their orientation and local brightness.

This method is based on checking whether a given data record can be classified as a member of a set of stored data records or not. Recognizing gestures and postures using template matching methods has two steps: Firstly, to create the templates (reference image of the gesture) by collecting data values for each gesture in the gesture set. The second part would be to find the gesture template most closely matching the current data record by comparing the current sensor reading with the given set. The main goal would be to construct a view-specific representation of the movement, where movement is defined as motion over time. The basic idea of this method is to build up a vector-image that can be matched against stored representations of known movements; this image is used as a temporal template.

Motion-Energy Images

Consider the example of someone sitting, as shown in Figure B.1:

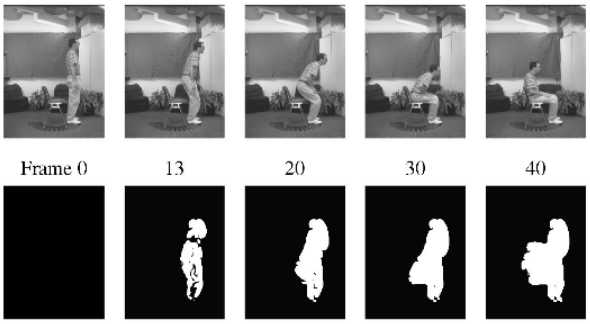

Figure B.1.: Example of someone sitting. Top row contains key frames. The bottom row is cumulative motion images starting from Frame 0.[16]

As it can be seen, the bottom row illustrates cumulative binary motion images computed from the start frame to the corresponding frame above. As expected, the sequence sweeps out a particular region of the image, and it's desired that the shape of that region (where there is motion) can be used to suggest both the movement occurring and viewing condition (angle). It's refer to these binary cumulative motion images as motion-energy images. Let $I(x, y, t)$ be an image sequence and let $D(x, y, t)$ be a binary image sequence indicating regions of motion, for many applications image differencing is adequate to generate D. Then, the binary motion-energy image $E_\tau(x, y, t)$ is defined:

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau-1} D(x, y, t - i)$$

As it can be seen, the duration τ is essential in defining the temporal extent of a movement. Fortunately, in the recognition section it's derived a backward-looking (in time) algorithm that dynamically searches over a range of τ.

Motion-History Images

To represent how (as opposed to where) the image is moving, it can be formed a motion-history image. In a motion-history image H_τ, pixel intensity is a function of the temporal history, and it's used a simple replacement of consecutive images and a decay operator:

$$\begin{cases} \tau, & if D(x, y, t) \\ \max(H_\tau(x, y, t-1) - 1), & otherwise \end{cases}$$

The result is a scalar-valued image where more recently moving pixels are brighter. One possible objection to the approach described here is that there is no consideration of optic flow, the direction of image motion. In response, it is important to note the relation between the construction of the motion-history image and direction of motion. Consider the movement example in Figure B.2.

Figure B.2.: Arms-up gesture and its MHI representation. [16]

Because the arms and legs are isolated components (they do not occlude other moving components), the motion-history image implicitly represents the direction of movement: The motion in the arm up position is "older" than the motion when the arms are down. For these types of articulated objects and for simple movements where there is not significant motion self-occlusion, the direction of motion is well-represented using the motion-history image. As motions become more complicated, the optic flow is more difficult to perceive, but is typically not lost completely.

Matching Temporal Templates

To construct a recognition system, it is needed to define a matching algorithm for the temporal template. Because it is being used an appearance-based approach, first it has to be defined the desired invariants for the matching technique and it is preferred to have a matching technique that is as invariant as possible to the imaging situation. Therefore, it has been selected a technique which is scale and translation invariant. There has to be a collection of training examples for each movement from a variety of viewing angles. Given a set of motion-energy images and motion-history images for each view/movement combination, it is computed statistical descriptions of these images using moment-based features. An acceptable choice would be around 7 Hu moments [56] which are known to yield reasonable shape discrimination in a translation and scale invariant manner.

For each view of each movement, a statistical model of the moments (mean and covariance matrix) is generated for both the motion-energy images and motion-history image. To recognize an input movement, a Mahalanobis distance [57] is calculated between the moment description of the input and each of the known movements. One disadvantage is that the Hu moments are difficult to reason and have a low intuition. Also, it can be noticed that the matching methods for the motion-energy images and motion-history image need not to be the same, in fact, given the distinction it has been made between where there is motion and how the motion is moving, it might be expected different matching criteria.

- Strengths

    1. Simplest technique to implement

    2. Accurate (for small set of postures)

    3. Requires only a small amount of calibration

- Weaknesses

    1. Not suited for hand gestures

    2. Does not work well for large posture sets due to over lapping templates

B.1.2. Feature Extraction Analysis

The goal of static gesture recognition is to classify the given gesture data represented by some features into some predefined finite number of gesture classes. The analysis of low-level information from the raw data in order to produce higher-level semantic information and use it to recognize postures and gestures is defined as Feature Extraction and Analysis. The system recognized these gestures with over 97% accuracy [58]. It is a robust way to recognize both simple postures and gestures and also complex ones as well. Feature extraction is part of the data reduction process and is followed by feature analysis. One of the important aspects of feature analysis is determining exactly which features are important [59]. Feature extraction is a complex problem in which the whole image or the transformed image is often taken as the input. The goal of feature extraction is to found the most discriminating information in the recorded images. Feature extraction operates on two-dimensional image arrays but produces a list of descriptions or a “feature vector” [60].

Mathematically, a feature is an n-dimensional vector with its components computed by some image analysis. The most commonly used visual cues are color, texture, shape, spatial information, and motion in video. For example, color may represent the color information in an image, such as color histogram, color binary sets, or color coherent vectors. The n components of a feature may be derived from one visual cue or from composite cues, such as the combination of color and texture [61]. The selection of good features is essential to gesture recognition because gestures are rich in shape variation, motion, and textures. Although body and hand postures can be recognized by extracting some geometric features such as fingertips, finger directions and hand or body contours, these features are not always available and reliable because of self-occlusion and lighting conditions. Finding a good data representation is a very domain specific and related to available measurements. Therefore, whole images or transformed images are taken as input, and features are selected implicitly and automatically by the classifier [62].

The problem of feature extraction can be descomposed in two steps: feature construction, and feature selection.

Feature construction

Human skills, which are often required to convert "raw" data into a set of useful features, can be complemented by automatic feature construction methods. In some approaches, feature construction is integrated in the modelling process whi may include:

- Standarization: Features can have different scales although they refer to comparable objects.

- Signal enhancement: The signal-to-noise ratio may be improved by applying signal or image-processing filters. These operations include baseline or background removal, de-noising, smoothing or sharpening.

- Extraction of local features: For sequential, spatial or other structured data, specific techniques like convolutional methods using hand-crafted kernels or syntactic and structural methods are used. These techniques are beyond the scope of this survey but it is worth mentioning that they can bring significant improvement.

- Linear and non-linear space embedding methods: When the dimensionality of the data is very high, some techniques might be used to project or embed the data into a lower dimensional space while retaining as much information as possible.

- Feature discretization: Some algorithms do not handle well continuous data. It makes sense then to discretize continuous values into finite discrete set. This step not only facilitates the use of certain algorithms, it may simplify the data description and improve data understanding.

Feature construction is one of the key steps in the data analysis process, largely conditioning the success fo any subsequent statistics or machine learning process.

Feature selection

Although feature selection is primarily performed to select relevant and informative features, it can have other motivation, including:

- General data reduction to limit storage requirements and increase algorithm speed.

- Feature set reduction to save resources in the next round of data collection or during utilization.

- Performance improvement, to gain in predictive accuracy.

- Data understanding to gain knowledge about the process that generated the data or simlpy visualize the data.

- Strengths

    1. Handles postures and gestures equally well.

    2. Uses layered architecture to analyse postures and gestures.

- Weaknesses

    1. Can be computationally expensive depending on how many features are extracted

B.1.3. Active Shapes Model

In this technique, a contour on the image that is roughly the shape of the feature to be tracked is used. After the hand contours are found out by a real-time segmenting and tracking system, a set of

feature points (landmarks) are marked out automatically and manually along the contour. A set of feature vectors will be normalized and aligned and then trained by Principle Component Analysis. Mean shape, eigenvalues and eigenvectors are computed out and composed of active shape model. The manipulation of contour is done by moving it iteratively toward nearby edges that deform the contour to fit the feature. Active Shape Model is applied to each frame and use the position of the feature in that frame as an initial approximation for the next frame. When the model parameter is adjusted continually, various shape contours are generated to match the hand edges extracted from the original images. The gesture is finally recognized after well matching.

Training set and landmarks computation

The first step in active shapes model is to build a training set from which the statistical properties of the class of objects will be learnt. Depending on the hand contour extraction on the first section, it's made up the hand-contour training sets that could be like it's shown in Figure B.3:

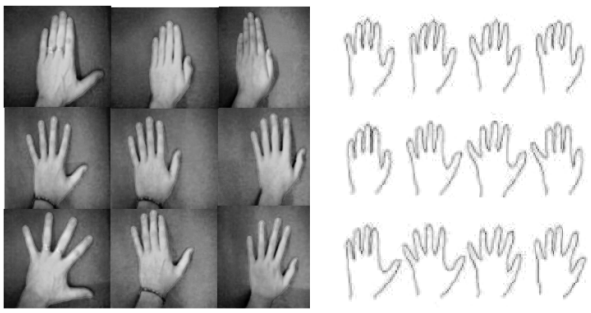

Figure B.3.: Images and contours from a training set. (Similar to [17])

In active shapes model, a shape is defined by a set of points called landmark. In order to define them, after the image has been segmented and its contour is extracted, there are taken the coordinates of the contour. Next, the curvatures of the contour are calculated. For each point i of the contour, it is considering the n contour points before and after it, and the parameter t is $2n + 1$. We interpolate x and y in function of the parameter by two polynomial functions. As a result, we obtain $x(t)$ and $y(t)$. We can then compute the curvature radius R of the parameter function defined by $(x(t)\ y(t))$ at the point i:

$$R = \frac{(x'^2 + y'^2)^{\frac{3}{2}}}{x'y'' - y'x''}$$

If R is below a predefined threshold, then the curvature is quite high at point i, and the point is retained as illustrates the Figure B.4.c. When more than a certain number of consecutive points are retained, the region of high curvature along the contour reaches, which are in this case the fingertips and the junctions of the fingers. Then, it's kept the median point of the consecutive high curvature points as a landmark as it can be seen in Figure B.4.d.

This method works well, and could help to find out the candidates of the landmarks, further, on the base of them, when it's removed and added some new landmarks manually. Thereby, the landmarks vector of the image is available, which means the hand shape can be characterized by a vector s containing the coordinates of its n landmarks.

$$s = (x_1 \quad y_1 \quad x_2 \quad y_1 \quad \dots \quad x_n \quad y_n)^T$$

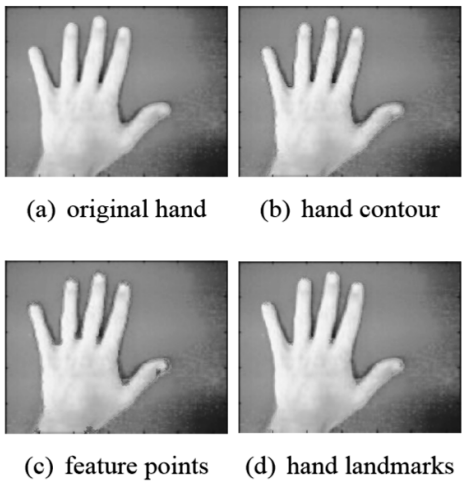

Figure B.4.: The labelled hand and its landmarks. (Similar to [17])

Each example in the training set then has to be labelled in a similar way. As a result, if the training set composed of m images labelled with n landmarks, a set of m $2n$-length vectors is then available:

$$_1s = (x_1 \quad y_1 1 \quad \dots \quad x_1 n \quad y_1 n)^T$$

$$s_2 = (x_2 1 \quad y_2 1 \quad \dots \quad x_2 n \quad y_2 n)^T$$

$$\vdots$$

$$s_m = (x_m 1 \quad y_m 1 \quad \dots \quad x_m n \quad y_m n)^T$$

The mean shape of the training set is defined as:

$$s = (x_1 \quad y_1 \quad x_2 \quad y_1 \quad \dots \quad x_n \quad y_n)^T$$

$$\bar{x}_i = \frac{1}{m} \sum_{j=1}^{m} x_j i$$

$$\bar{y}_i = \frac{1}{m} \sum_{j=1}^{m} y_j i$$

Align the shape

The position, scale and rotation is different between the extracted body shape, so it will be needed to align the shape sets. The Figure B.5 shows the flowing chain [63].

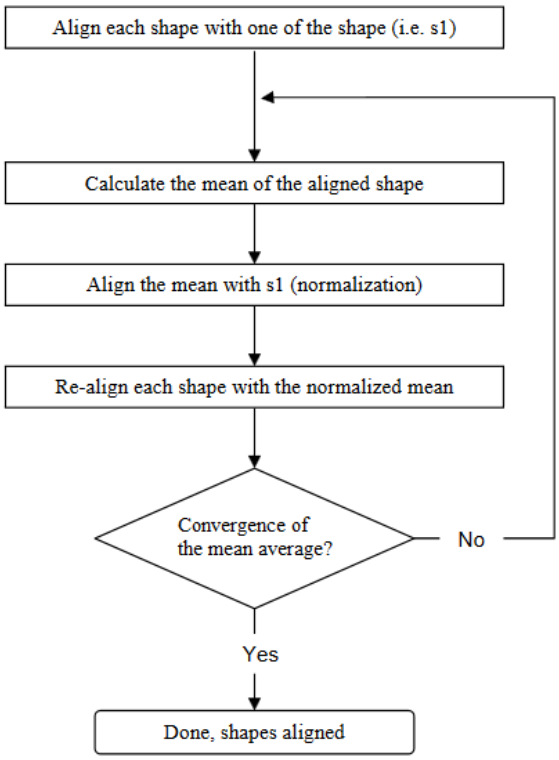

Figure B.5.: Flowing chart of aligning the shapes.

Shape alignment is to apply a transformation to each shape such that it becomes as similar as possible to the others in the sense of the minimization of a least-square function. The transformation used is a similarity (translation, scaling and rotation are allowed), so it's the method to find the parameters of the similarity that best aligns a shape with another.

- Strengths

  1. Allows real time recognition

  2. Handles both hand postures and gestures

- Weaknesses

  1. Tracks only the open hand

B.1.4. Principal component analysis

A statistical technique for reducing the dimensionality of a data set in which there are many interrelated variables is called Principal Component Analysis while retaining variation in the dataset [64].

Reduction of data set is made by transforming the old data to a new set of variables that are ordered so that the first few variables contain most of the variation present in the original variables. The original

data set is transformed by computing the eigenvectors and eigenvalues of the data set's covariance matrix. The eigenvector with the highest eigenvalue holds the highest variance, the eigenvector with the second highest eigenvalue holds the second highest variance, and so on.

Reduction of data set is made by transforming the old data to a new set of variables that are ordered so that the first few variables contain most of the variation present in the original variables. The original data set is transformed by computing the eigenvectors and eigenvalues of the data set's covariance matrix. The eigenvector with the highest eigenvalue holds the highest variance, the eigenvector with the second highest eigenvalue holds the second highest variance, and so on.

This system needs of sets of training images to generate a posture classifier that is then used to classify postures in real time. Each set of training images can be considered a multivariate data set: each image consists of N pixels and represents a point in N-dimensional space. In order for Principal Component Analysis to work successfully, there must be little variance in at least one direction and whatever variance exists should not be meaningful. This recognition system works well but there is little indication that Principal Component Analysis compresses the data set significantly beyond a naive approach.

Another important issue when dealing with image data is that it is highly sensitive to position, orientation, and scaling of the body in the image. Principal Component Analysis cannot transform two identical postures with different body sizes and positions to the same point. So each image has to be normalized to centre the body and rotate it so that its major axis is vertical, and scale it to fit the gesture image.

The posture classifier, is a transformation matrix containing the results of the Principal Component Analysis performed on all the images in the training set. After the transformation matrix has been calculated, the number of principal components is reduced by discarding the least important ones. The common approach is simply to keep the principal components with the N highest eigenvalues. Other information such as the number of posture classes, their mean, and their variance in the reduced data set can be used to choose the principal components.

A Bayes classifier is then used to recognize postures from the reduced set of principal components. Such parameters as image resolution and number of training images are also important components of Principal Component Analysis in a vision-based solution and can be modified to improve recognition results [65].

An example could be two different gestures (e.g. raising the hand –class A– and punching –class B–) that have been recorded several times with different people and different environments and have been classified into two different classes in an image database. It turns out that class A patterns tend to spread in a different area from class B patterns. The straight line seems to be a good candidate for separating the two classes. Now, it has been given a new image with a region in it and it has to be classified. The, it has to be measured the same features that were measured with the previous training set and the standard deviation in the region of interest in order to plot the corresponding point as shows the star in Figure B.6.

Then it is sensible to assume that the unknown pattern is more likely to belong to class A than class B. The measurements used for the classification, the mean value and the standard deviation in this case, are known as features. In the more general case n features $x_i, i = 1, 2, ..., n.$ are used and they form the feature vector where $x = [x_1, x_2, ..., x_n]^T$ where T denotes transposition. Each of the feature vectors identifies uniquely a single pattern (object). The straight line in Figure B.6 is known as the decision line, and it constitutes the classifier whose role is to divide the feature space into regions that correspond to either class A or class B. If a feature vector x, corresponding to an unknown pattern, falls in the class A

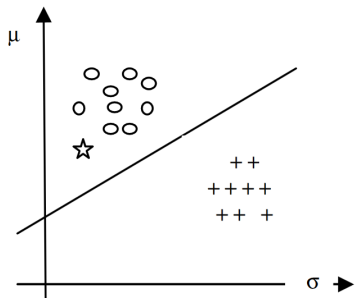

Figure B.6.: Plot of the mean value versus the standard deviation for a number of different images originating from class A (o) and class B (+). [1]

region, it is classified as class A, otherwise as class B. The patterns (feature vectors) whose true class is known and which are used for the design of the classifier are known as training patterns (training feature vectors).

- Strengths

  1. Can recognize on the order of 25 to 35 postures[65].

- Weaknesses

  1. Requires training by more than one person for accurate results and user independence [65].

  2. Requires normalization to keep images consistent.

B.1.5. Causal Analysis

This technique attempts to extract information from a continuous image stream by using high-level knowledge about actions in the scene and how they relate to one another and the physical environment. Examples of causal data (e.g. the underlying physics of the scene) include rigidity, mass, friction, balance, and work against gravity. By using knowledge about body kinematics and dynamics, features recovered from the video stream can be used to identify gestures based on human motor plans. The system captures information on shoulder, elbow and wrist joint positions in the image plane. From these positions, the system extracts a feature set that includes wrist acceleration and deceleration, work done against gravity, size of gesture, area between arms, angle between forearms, nearness to body, and verticality. Gesture filters normalize and combine the features and use causal knowledge of how humans interact with objects in the physical world to recognize gestures such as opening, lifting, patting, pushing, stopping, and clutching.

- Strengths

  1. Uses information about how humans interact with the physical world to help identify gestures.

- Weaknesses

1. Uses only a limited set of gestures.

2. Does not use hand orientation and position data or finger data.

3. Does not run in real time.

Appendix C

Evaluation questionnaire

Evaluation bachelor thesis: Kinect sensor in virtual weather TV studio

PERSONAL DATA	PARTICIPANT N°:
NAME:	NATIONALITY:
AGE:	OCCUPATION:
HEIGHT:	

HAVE YOU MODERATION / PRESENTATION EXPERIENCE?

YES ☐

NO ☐

USING THE SENSOR

	Not at all		Medium		Perfectly
1. Was comfortable to perform gestures in front of the sensor?	☐	☐	☐	☐	☐
2. Could the gestures be learned easily?	☐	☐	☐	☐	☐
3. How accurate has the gesture recognition felt?	☐	☐	☐	☐	☐
4. Did you notice errors in the gesture recognition?	☐	☐	☐	☐	☐
5. Were the interactions easy to carry out?	☐	☐	☐	☐	☐
6. How intuitive were the gestures?	☐	☐	☐	☐	☐
7. How easily could the gestures sequence be learned?	☐	☐	☐	☐	☐
8. Were the sounds as feedback helpful?	☐	☐	☐	☐	☐
9. Was there a smooth tracking (precise and continuous following of the movements) ?	☐	☐	☐	☐	☐
10. How much have disturbed the errors?	☐	☐	☐	☐	☐

C. Evaluation questionnaire

IS FURTHER FEEDBACK NECESSARY?

YES ☐

NO ☐

IF SO, WHAT FEEDBACK MIGHT BE HELPFUL?

WAS THERE ANY VISIBLE DELAY?

YES ☐

NO ☐

IF SO, WHICH KIND AND WHEN DID YOU EXPERIENCED THIS DELAY?

WHICH INTERACTION MADE MORE SENSE TO YOU? (CHOOSE MAXIMUM 3)

☐ STARTING

☐ SELECTING BETWEEN THE BUTTONS

☐ CHANGING BETWEEN DISPLAYS

☐ SCROLLING THE MAP

☐ ZOOMING THE MAP

WHICH GESTURE WAS BEST DETECTED?

☐ START ☐ ZOOM IN

☐ POINT UP ☐ ZOOM OUT

☐ POINT DOWN ☐ SCROLL UP

☐ POINT MIDDLE ☐ SCROLL DOWN

VIRTUAL STUDIO

	Not at all		Medium		Perfectly
1. Were all the imaginable possible interactions covered?	☐	☐	☐	☐	☐
2. Were the interactions as you expected to be?	☐	☐	☐	☐	☐
3. How useful was the Kinect sensor in the weather studio?	☐	☐	☐	☐	☐
4. Did the gestures disturb you?	☐	☐	☐	☐	☐

COULD THE KINECT SENSOR BE USED IN A REAL LIVE TV BROADCAST PRODUCTION SCENARIO?

YES ☐

NO ☐

DO YOU THINK THERE ARE OTHER POSSIBLE VIRTUAL TV STUDIO SETS FOR THE KINECT SENSOR?

YES ☐

NO ☐

IF SO, IN WHICH OTHER VIRTUAL SETS THE KINECT SENSOR COULD BE USED?

ARE THERE ANY OTHER KIND OF SET THAT COULD BE MORE SUITABLE?

YES ☐

NO ☐

HOW WAS THE EXPERIENCE OF USING THE INTERACTIVE SYSTEM? WAS IT AS YOU EXPECTED TO BE?

WERE THERE ANY VISIBLE ERRORS?

YES ☐

NO ☐

IF SO, WHICH ERRORS HAVE BEEN DETECTED?

THAT WAS ALL. THANK YOU FOR YOUR PARTICIPATION!

Appendix D

Evaluation results

In the next table are exposed the questions that were made to the volunteers after testing the Kinect system as shown in Appendix C. To the following questions have been attached the 15 answers of each of the user in a scale from 1 (not at all) to 5 (perfectly). In the third column are shown the arithmetic mean values for each question.

QUESTION	ANSWER															MEAN
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	
Was comfortable to perform gestures in front of the sensor?	3	5	5	5	4	4	5	5	4	4	5	4	5	3	4	4.3
Could the gestures be learned easily?	3	5	5	5	3	3	4	5	4	5	4	3	4	4	5	4.1
How accurate has the gesture recognition felt?	3	4	3	4	3	3	4	3	3	4	4	3	4	4	4	3.5
Did you notice errors in the gesture recognition?	4	4	4	4	4	3	3	3	3	4	5	4	3	3	1	3.5
Were the interactions easy to carry out?	4	5	5	5	5	3	5	4	4	5	5	4	4	4	5	4.5
How intuitive were the gestures?	4	5	5	5	3	5	5	5	5	5	5	5	5	5	5	4.8
How easily could the gestures sequence be learned?	3	5	5	5	4	4	4	4	5	4	4	4	4	4	5	4.3
Were the sounds as feedback helpful?	5	5	5	5	4	5	5	5	5	5	5	5	5	5	5	4.9
Was a smooth tracking?	3	4	3	4	3	4	4	3	3	4	3	3	4	3	4	3.5
How much have disturbed the errors?	4	2	4	4	3	3	3	3	3	4	4	3	3	3	1	3.1
Were all the imaginable possible interactions covered?	3	5	5	5	5	4	5	5	5	5	5	5	5	4	5	4.7
Were the interactions as you expected to be?	3	5	4	4	5	4	5	5	5	4	4	4	4	4	5	4.3
How useful was the Kinect sensor in the weather studio?	4	5	5	5	5	5	5	5	4	5	5	4	5	4	5	4.7
Did the gestures disturb you?	3	1	1	1	1	3	2	2	1	2	1	2	1	2	1	1.6

Bibliography

[1] José Ignacio Herranz Herruzo. *Reconocimiento y Clasificación*. Universidad Politécnica de Valencia, November 2015.

[2] William T. Freeman and Michal Roth. Orientation histograms for hand gesture recognition. Technical report, Mitsubishi Electric Research Labs., 201, 213.

[3] Jakub Segen and Senthil Kumar. Gesture VR: vision-based 3D hand interace for spatial interaction. In *ACM Multimedia Conference*, pages 455–464, 1998.

[4] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 21(8):745–758, 2003.

[5] Nguyen Dang Binh, Enokida Shuichi, and Toshiaki Ejima. Real-time hand tracking and gesture recognition system. In *Proceedings of International Conference on Graphics, Vision and Image Processing (GVIP-05*, pages 362–368, 2005.

[6] J. Appenrodt, S. Handrich, A. Al-Hamadi, and B. Michaelis. Multi stereo camera data fusion for fingertip detection in gesture recognition systems. In *Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of*, pages 35–40, Dec 2010.

[7] Larry Li. Time-of-flight camera–an introduction. *Technical White Paper, May*, 2014.

[8] David Fofi, Tadeusz Sliwa, and Yvon Voisin. A comparative survey on invisible structured light. In *Electronic Imaging 2004*, pages 90–98. International Society for Optics and Photonics, 2004.

[9] Christian Moore. Kinect 2: What does it sense? `http://nuigroup.com/forums/viewthread/14981/#72710`, 2013. Last update 21 May 2013.

[10] Kinect. `http://www.derivative.ca/wiki088/index.php?title=Kinect`, 2015. Last update 20-11-2015.

[11] Microsoft©. Visual Gesture Builder: a Data-Driven Solution to Gesture Detection. Technical report, 2014. Creation date: 2013-09-18, Modification date: 2014-07-20.

[12] Marina Ballester-Ripoll, Jens Herder, Philipp Ladwig, and Kai Vermeegen. Comparison of two gesture recognition sensors for virtual tv studios. 2016.

[13] Feedforward neural networks. `http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__What_is_a_feedforward_ne.html`, 2004. Created the 26-04-2004.

[14] Inc StatSoft. Electronic statistics textbook. *Tulsa, OK: Electronic Statistics Textbook*, 2006.

[15] DR NAPOLEON H REYES. Real-time hand gesture detection and recognition using simple heuristic rules.

[16] Aaron F Bobick and James W Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, 2001.

[17] Nianjun Liu and Brian C Lovell. Hand gesture extraction by active shape models. In *Digital Image Computing: Techniques and Applications (DICTA'05)*, pages 10–10. IEEE, 2005.

[18] Hyung-Ji Lee and Jae-Ho Chung. Hand gesture recognition using orientation histogram. In *TENCON 99. Proceedings of the IEEE Region 10 Conference*, volume 2, pages 1355–1358 vol.2, Dec 1999.

[19] J. Triesch and C. von der Malsburg. Robust classification of hand postures against complex backgrounds. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 170–175, Oct 1996.

[20] José Manuel Iñesta Quereda and Luisa Micó, editors. *Pattern Recognition in Information Systems, Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems, PRIS 2002, In conjunction with ICEIS 2002, Ciudad Real, Spain, April 2002*. ICEIS Press, 2002.

[21] Chin chen Chang, Jiann jone Chen, Wen kai Tai, and Chin chuan Han. New approach for static gesture recognition *.

[22] M. A. Amin and H. Yan. Sign language finger alphabet recognition from gabor-pca representation of hand gestures. In *2007 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2218–2223, Aug 2007.

[23] Shikha Gupta, Jafreezal Jaafar, and Wan Fatimah Wan Ahmad. Static hand gesture recognition using local gabor filter. *Procedia Engineering*, 41:827 – 832, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[24] Chao Hu, M. Q. Meng, P. X. Liu, and Xiang Wang. Visual gesture recognition for human-machine interface of robot teleoperation. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1560–1565 vol.2, Oct 2003.

[25] L. Yun and Z. Peng. An automatic hand gesture recognition system based on viola-jones method and svms. In *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on*, volume 2, pages 72–76, Oct 2009.

[26] G. Lei, X. h. Li, J. l. Zhou, and X. g. Gong. Geometric feature based facial expression recognition using multiclass support vector machines. In *Granular Computing, 2009, GRC '09. IEEE International Conference on*, pages 318–321, Aug 2009.

[27] Z. Ren, J. Yuan, J. Meng, and Z. Zhang. Robust part-based hand gesture recognition using kinect sensor. *IEEE Transactions on Multimedia*, 15(5):1110–1120, Aug 2013.

[28] J. M. Rehg and T. Kanade. Digiteyes: vision-based hand tracking for human-computer interaction. In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, pages 16–22, Nov 1994.

[29] A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, page 478 Vol. 1, 1999.

[30] Prashan Premaratne, Farzad Safaei, and Quang Nguyen. Moment invariant based control system using hand gestures. In *Intelligent Computing in Signal Processing and Pattern Recognition*, pages 322–333. Springer, 2006.

[31] P.Progscheba W.Vonolfen M.Flasko, J.Herder. Heterogeneous binocular camera-tracking in a virtual studio. *GI-VRAR*, 2011.

[32] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Patrice Horaud. *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media, 2012.

[33] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Adv. Opt. Photon.*, 3(2):128–160, Jun 2011.

[34] Joaquim Salvi, Jordi Pages, and Joan Batlle. Pattern codification strategies in structured light systems. *Pattern recognition*, 37(4):827–849, 2004.

[35] Achuta Kadambi, Ayush Bhandari, and Ramesh Raskar. 3d depth cameras in vision: Benefits and limitations of the hardware. In *Computer Vision and Machine Learning with RGB-D Sensors*, pages 3–26. Springer, 2014.

[36] Heuristic based gesture detection techniques. (kinect content at gamefest 2011).

[37] Gesture detection using machine learning. (kinect content at gamefest 2011).

[38] Microsoft. Visual Gesture Builder: Overview. `https://msdn.microsoft.com/en-us/library/dn785529.aspx`, 2014. Created the 15-10-2014.

[39] Robert E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[40] Andy Liaw and Matthew Wiener. Classification and regression by random forest. *R news*, 2(3):18–22, 2002.

[41] Yi-Wei Chen and Chih-Jen Lin. Combining svms with various feature selection strategies. In *Feature extraction*, pages 315–324. Springer, 2006.

[42] Jonathan Simsch and Jens Herder. Spiderfeedback: visual feedback for orientation in virtual tv studios. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, page 12. ACM, 2014.

[43] Sven Kratz and Rafael Ballagas. Unravelling seams: improving mobile gesture recognition with visual feedback techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 937–940. ACM, 2009.

[44] Andy Cassidy, Dan Hook, and Avinash Baliga. Hand tracking using spatial gesture modeling and visual feedback for a virtual dj system. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 197. IEEE Computer Society, 2002.

[45] Kai Vermeegen. Interaktionen mit virtuellen Sets innerhalb eines virtuellen Studios mittels eines Gestensteuerungs-armbandes. Bachelorthesis, FH Duesseldorf, 2016.

[46] Hermann Fürntratt and Helmut Neuschmied. Evaluating pointing accuracy on kinect v2 sensor. In *International Conference on Multimedia and Human-Computer Interaction (MHCI)*, pages 124–1, 2014.

[47] Majdi Alnowami, Afzal Khan, Ali H Morfeq, Nazeeh Alothmany, and Ehab A Hafez. Feasibility study of markerless gait tracking using kinect. *Life Science Journal*, 11(7), 2014.

[48] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[49] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[50] Ben Kröse, Ben Krose, Patrick van der Smagt, and Patrick Smagt. An introduction to neural networks, 1993.

[51] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[52] S.K. Mathanker, P.R. Weckler, T.J. Bowser, N. Wang, and N.O. Maness. Adaboost classifiers for pecan defect classification. *Comput. Electron. Agric.*, 77(1):60–68, June 2011.

[53] Chieh-Chih Wang and Co-Chih Wang. Hand posture recognition using adaboost with sift for human robot interaction. In *Proceedings of the International Conference on Advanced Robotics (ICAR'07)*, Jeju, Korea, August 2007.

[54] Ada Boost. `http://www.nickgillian.com/wiki/pmwiki.php/GRT/AdaBoost`, 2014. Created the 17-04-2014.

[55] Björn Stenger. Template-based hand pose recognition using multiple cues. In *Computer Vision–ACCV 2006*, pages 551–560. Springer, 2006.

[56] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

[57] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[58] Yinghui Zhou, Lei Jing, Junbo Wang, and Zixue Cheng. Analysis and selection of features for gesture recognition based on a micro wearable device. 2012.

[59] Scott E. Umbaugh. *Computer Vision and Image Processing: A Practical Approach Using Cviptools with Cdrom*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1997.

[60] G. J. Awcock and R. Thomas. *Applied Image Processing*. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[61] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3):245–268.

[62] Ying Wu and Thomas S. Huang. *Gesture-Based Communication in Human-Computer Interaction: International GestureWorkshop, GW'99 Gif-sur-Yvette, France, March 17-19, 1999 Proceedings*, chapter Vision-Based Gesture Recognition: A Review, pages 103–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[63] Finding optimal rotation and translation between corresponding 3D points. `http://nghiaho.com/?page_id=671`, 2014. Last update 10-05-2013.

[64] N. H. Dardas and E. M. Petriu. Hand gesture detection and recognition using principal component analysis. In *2011 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings*, pages 1–6, Sept 2011.

[65] Henrik Birk, Thomas B. Moeslund, and Claus B. Madsen. Real-time recognition of hand alphabet gestures using principal component analysis. In *In 10th Scandinavian Conference on Image Analysis*, 1997.