Tiago Pinheiro de Melo

Gesture Recognition for Natural Interaction with Applications

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Julho de 2012

Tiago Pinheiro de Melo

Gesture Recognition for Natural Interaction with Applications

Relatório de Estágio submetido à
Faculdade de Ciências da Universidade do Porto
para obtenção do grau de Mestre em
Engenharia de Redes em Sistemas Informáticos

Orientador: António Santos
Co-orientador: Alípio Jorge

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Julho de 2012

Acknowledgements

To Fraunhofer Portugal for the opportunity and for providing the necessary means to complete this research.

António Santos who supervised me and always keeping me in the right direction. He advised me with important research lines that truly helped on the progression of the work and gave important insights on how to solve problems.

Alípio Jorge for giving me important insights and useful recommendations to complete my research.

To my family for their love, specially my parents, who did everything during these years so I could be here finishing my thesis, and that allowed me to become who I am today.

For the support and encouragement, a special thanks to my girlfriend and friends.

Resumo

O reconhecimento de gestos tem vindo a tornar-se uma tecnologia com a qual os humanos se têm cada vez mais vindo a familiarizar, permitindo que as interfaces das máquinas se consigam afirmar cada vez mais nas nossas casas. Com o avanço da tecnologia surgiram os sensores 3D que hoje em dia são usados principalmente em jogos. Os idosos no entanto, ainda enfrentam diversos obstáculos numa sociedade cada vez mais tecnológica e, devido a esse facto, estão em desvantagem e é necessário que investigadores assegurem que este público não seja excluído dos benefícios que a tecnologia pode proporcionar.

O principal objectivo deste trabalho é desenvolver um *driver* que consiga processar todos os dados enviados pelo sensor Kinect da Microsoft, identificando gestos para depois poder controlar um sistema com o qual seja possível interagir apenas com eventos de teclado. Definiu-se como objectivo e caso de uso para este trabalho poder controlar uma set-top box desenvolvida no âmbito de outro projecto. Este sistema possui uma interface gráfica simples de navegação de menus, proporcionando uma facil interação através de gestos. Os gestos definidos têm que realizar eventos, que deverão ser enviados através da camada de *input* do respectivo sistema operativo.

Para atingir este propósito, numa fase inicial foi feito um estudo aprofundado sobre as frameworks que dão suporte ao Kinect em Linux. Com base nesse estudo, procedeu-se à realização do *driver* definindo os gestos de maneira intuitiva e simples para o público-alvo em causa. Com base nos testes que foram realizados em idosos, conseguiu-se depois aproximar os gestos dos testes e implementá-los no *driver Kinteract*. Para finalizar, é possível ainda configurar o *Kinteract* com diversos parâmetros de afinação, assim como os eventos a enviar por cada gesto reconhecido. well the events to sent by each gesture recognized.

Abstract

Gesture recognition has become a very familiar technology for humans, allowing the machine interfaces to be much more usable in our houses. With the technological progress emerged the 3D sensors which nowadays have been used primally in videogames. Old people, however, are still facing many obstacles an even more techonological society and, due to this fact they are in disadvantage and it's necessary that researchers ensure that these people don't be excluded from the benefits provided by this techonology.

The main goal of this work is to develop a driver that can process all the data sent from Microsoft Kinect sensor, identifying gestures and being able to control a system, which possible only interact with keyboard events. We defined as objective and use case for this work, be able to control a set-top box developed under another project. This system has a simple graphical interface of navigation menus, providing an easy interaction through gestures. Those defined gestures have to be translated into keyboard events, which should be sent through the input layer of the operating system.

To meet this goal, in an initial deep study was made around the frameworks that support Kinect on. Based on that study, we proceeded with the development of Kinteract to completion of the driver, defining the gesture in an intuitive and simple way for the target audience. From the tests that we have done in old people, then we we could fine-tune the recognized gestures based on the results of the tests and implement them in Kinteract. Finally, it's still possible configurate the Kinteract with several relaying parameters, as well the events to sent by each gesture recognized.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation/Context

Nowadays we are living in the "digital" era. Our lifes were invaded by several technologies, that the majority have the purpose to ease and simplify our lives in several aspects and areas. Technology usually meets people but one of the first points is to define the main target audience, and then adapt it to work properly, providing all innovation to all the people that will use it.

The keyword for this is human interaction. This term is important when we realize any technology that will interact with people. Nowadays we have technologies that don't reach everyone because not all consumers are skilled. The researchers should be aware that these innovations must be accessed by everyone regardless of their technical knowledge.

There are several motion sensors available nowadays in the market that can be used as natural user interface for applications and games, through the use of gestures and spoken commands. The Kinect sensor by Microsoft and the Asus Xtion Pro sensor are two examples. These sensors were and are mainly used for games but the scientific community has been finding many areas where this kind of devices can be of major utility. One of those areas is Ambient Assisted Living.

Due to the frequent inability of most older adults in interacting with new devices and in order to promote healthy physical movements, Fraunhofer AICOS is now building a solid knowledge base in working with this kind of sensors in order to use them as interfaces for

many applications in this area. The main purpose of this project is to allow it to become true for older people, because this is the main target audience of this project.

1.2 Objectives

The primary goal of this thesis is to have the Kinect sensor integrated with a set-top box that has been developed in previous projects by Fraunhofer, featuring a simplified user interface, and provide the user with the ability to interact with that interface by using simple gestures instead of the usual remote control. For that, we have to develop a daemon (for the Linux OS) that processes data sent by the sensor and identifies simple pre-defined gestures that should be translated to input events (like the cursor keys, escape, enter, etc.) which then should be sent to the Linux input layer.

To be able to successfully finish this project we had to study the protocol used by this kind of sensors and the existing drivers and frameworks (OpenKinect, OpenNI/NITE, etc.) that aid in developing applications that use gesture recognition.

1.3 Project Description

The aim of this project is to allow a broad use of the Kinect sensor with any existing application. In our case, we want to use it in the context of Ambient Assisted Living, a very important research area with a high social impact. We have developed a driver/daemon that processes raw input from the sensor Kinect from Microsoft, identifies simple gestures, and injects events into the Linux input layer. This program will be integrated in the operating system of a set-top box and the user must be able to fully control it using gestures only.

1.4 Thesis outline

This thesis is structured in six chapters. This chapter introduces the context of this research together with its goals and contribution and this thesis outline.

Chapter two develops an understanding in areas where this technology applies. It starts

with a review about the evolution of games since the introduction of new interacting and controlling devices. Next, a brief explanation of Kinect, their components and characteristics that are part of it. Next, and not least important we show examples of areas where Kinect already has been used, showing the advantages of this technology in different areas. Finally, we explain the frameworks that allow the development of applications using Kinect and expose their main features and advantages/disadvantages.

The third chapter discusses technical aspects about the Kinect. We explain the protocol that Kinect uses, showed how the components communicate with the computer and how sent from the sensor.

The solution proposal and its features are presented in chapter four. Chapter four also shows the high level design the project, how we use the frameworks, and also identify which are the main modules that form Kinteract.

In chapter five there's a deeper explanation of some implementation details related to Kinteract. This chapter also contains an explanation of the uinput module and library libxml, since both are very important for Kinteract, being a generic driver for interacting with applications in a Linux operating system.

In chapter six we show the results of the tests that we made with five persons, and explain the purpose of these tests that helped us to define the gestures for the main target audience: old people.

Finally, chapter seven provides some conclusions of our work and discusses the gestures which we were expecting, and that can be improved on future research and works.

1.5 Achievements

In this work we were able to implement the gestures defined by the old, in driver Kinteract, with the help of frameworks that have already tools to detect movements like cursor keys. The movements to control the events "Left", "Right", "Up", "Down" and "Ok" are well defined, because they correspond to the gestures made by the older people in the tests. The "Back" gesture was more confusing for the old people, but we were able to find a pattern in the tests and tried to implement it in Kinteract. Our current implementation has some identified limitations that will be dealt with in future developments.

Chapter 2

Natural Interaction: State of the Art

2.1 Background

In 1946, the introduction of the first digital computer, named ENIAC, was the starting point for an amazing technological development, encouraging several forms of human-computer interaction.

Human-computer interaction has been evolving all over the history, in order to help people to complete basic tasks in a safe and efficient mode. Despite the evolution of computer technology, this is still an always evolving tool controlled by man. The history of interface and interaction design at the beginning was only for a few persons, now the developers have the responsibility of increasing the simplicity of the systems for everyone to be able to use, in everyday life. Nowadays, in many cases, people use too complex machines and interfaces to accomplish very simple tasks. One reason for this is that common interfaces are often based on abstract approaches.

The gestures and the expressions of the people are targets of investigation and the main purpose is development systems that have to process and recognize gestures in order to use any device without the need to learn any instruction. This same concept should be intuitive, easy and attractive, for facilitating interaction with any graphical environment.

The success of this system depends on the impact created on people, evidencing that they are dealing with something real, not any abstract thing.

2.1.1 Basic concepts

Natural Interaction

Natural interaction research is a point of intersection of different areas like computer science, mechanical engineering and creative design. An example of this was Leonardo da Vinci that represents a model of researcher across different disciplines, showing the benefits of knowing different fields, and as a result he was an artist, a scientist and a technologist at the same time.

In other words, Valli [37] describes the natural interaction as a way to allow the manipulation of an environment, in which the need for learning interaction is minimal, that is, the interaction metaphor used to be so simple that anyone could use, requiring a minimal training.

Natural Interaction is an art for we can guide certain behaviors through stimuli, movements or gestures. The concept of natural interaction being similar to augmented reality should be as intuitive as possible to be positively accepted by people. To handle a natural environment, augmented reality must have ways to interact naturally.

Augmented Reality

Augmented Reality refers to a simple combination of virtual environments and virtual reality. Virtual environments are environments or systems in which users interact with each other. [32] Virtual reality is in an artificial environment that is created with software. [31]

Augmented reality can be found in several areas. For example, games were determinant to the advancement of technology, producing an evolutionary impact on intuition and perception of the games for the target audience, as well as others (medical, entertainment, military training), because technology is so pervasive throughout the medical field that it isn't surprising that this domain is viewed as one of the more important for augmented reality systems. [38]

2.2 The evolution of consoles

The evolution of console games in augmented reality started in 1988 with the Power Pad, a Super Nintendo accessory. Its initial purpose was to make the player lose weight with a carpet containing twelve pressure sensors working as buttons under the feet. However, it didn't work as expected because, among other factors, only eleven games were released.

In 1993, Sega developed the first sensory control called Sega Activator, which contained an octagon that was put on the floor, with infra-red sensors, having limitations in some actions like giving a punch, for example in Street Fighter 2 or Mortal Kombat 2, because it wasn't enough to represent the gesture of the punch, the user had to do the move in the place that represented the punch button.

Dance dance revolution was started in Japan on 1998. The game worked as the current Guitar Hero, a list of movements slides across the screen, and with his feet, the player should compress the buttons following the sequence showed on the screen. The carpet came from Dreamcast, through PSX, PC, PS2, Game Cube, Wii, and Xbox coming in the next generation console Xbox 360 a few years later.

Returning to the Dreamcast, there was a game called "Samba de Amigo", that had success at the time that it was released (in 2000) and for adding fun, the game came with "maracas controls", a kit including a carpet, allowing to detect when the controls were going down or up.

PlayStation 2 also had another motion related accessory, called Eye Toy. It was launched in 2003 and looked like a webcam. Besides taking pictures, the main idea allowed a basic recognition of movements provide, the control of the game without the need of others accessories.

This brings us to 2006 with the release of the Nintendo Wii, a console essentially done to put people moving by the use of several controls, including an infrared camera that is able to capture the movements of the players, making the games more funny and entertaining. This console is more focused on entertainment, and not in mind-blowing graphics, a strategy that managed the console to conquer the whole family, being the best-selling console of this generation.

With all the success of the Nintendo Wii, their main competitors Sony and Microsoft ran after, and launched there own motion detectors. Sony came first and brought to market

the Ps Move.

The PlayStation Move works by using a combination of accelerometers, gyrometers, and magnetometers. Sony determined that the sensors alone weren't enough to accurately track movement, and as a result, the lit bulb on top of the controller works in pair with the PlayStation Eye to help do so. The colors of the bulbs can also change shade to help the PS Eye better track movement in case a particular room environment has similar color schemes.

With the technology constantly growing came the smartphones, tablets, or touch interfaces that in some way also projected the evolution of games, specially because they are a little similar to the Kinect, in the way that recognizes touch gestures to control devices. More recently in June 2010, Microsoft announced "Project Natal", an earlier name for the later called Kinect. It is a peripheral originally developed for the Xbox 360 by Microsoft, which now has been allowing the development of programs to perform communication through body gestures and sound . Then in November 2010, the Microsoft Kinect was released in North America, and, shortly after the release, an open source community was adapting its operation for personal computers.

This equipment is capable of mapping the environment in three dimensions, to film the players and receive orders through their microphones. In a simple definition, the Microsoft Kinect gives directions to the machine, and with this, the hackers quickly found out that it was a totally independent and alternative device.

Although being a revolutionary idea, this device isn't alone in the market, because in February 28, 2011 has been made the presentation of the Wavi Xtion from Asus at CeBIT (technology fair in Hannover, Germany). This device is similar to Kinect, but unlike the Microsoft device in the beginning when it was built, Wavi Xtion isn't restricted to games and can be used as a media center.

This device consists of two parts, a media center (which can transmit the contents of your home computer through streaming) and the sensor. Both must be connected to the TV, and some gestures would open photo albums, watch videos or play compatible games.

2.3 Microsoft Kinect

The Microsoft's Project Natal, as announced at 2009 in Electronic Entertainment Expo[1].This is the code name that Microsoft gave to the Kinect, and in same year was rumored that the Project Natal would be associated with the release of Xbox 360 console. [15]

The Microsoft Kinect is a device sold by Microsoft initially to be used on the Xbox 360, a video game console that, with this extra device, allows the user to interact naturally with it, through body gestures and voice commands, with the purpose of entertainment.

This accessory is interesting because it features a depth sensor, microphone and sensor space. It can read three-dimensional spaces and recognize forty eight points of articulation in people and so it doesn't need any contro or point of light in the hand. With two cameras (one is infrared, with two lenses) it reads the player movements and transfers them to the game. The Kinect has microphones that allow some degree of control through the use of voice. Its technical specifications include:

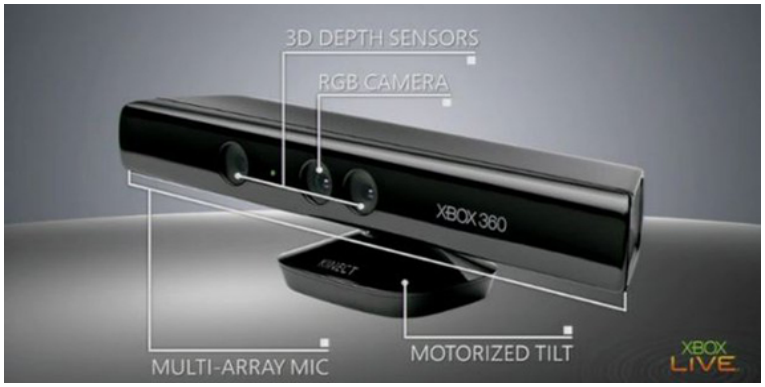

Figure 2.1: The Kinect. [7]

- RGB Camera [29]: This video camera helps in facial recognition and other detection features by detecting three color components: red, green and blue. Microsoft calls this an "RGB camera" referring to the color components it detects.

- 3D Depth Sensors [29]: It includes a projector that bounces out a laser across the

[1] **E3** - Electronic Entertainment Expo, commonly known as E3, is an annual trade fair for the computer and video games industry, realized in Los Angeles Convention center.

entire field of play and an IR VGA camera which picks it up to separate the player from the rest of the room. It gets back IR noise measured in varying color depending on how close it is to the system. Bodies appear as a bright shade of red, green and things further away appear in gray. The software takes this image and put it through filters so that Kinect can discover what is a person and what isn't. It then converts this information into a skeleton with moving joints.

- Engine [29]: This was necessary since not every room is the same size and not every TV is exactly at the same height. It can tilt up or down thirty degrees. The motor also operates the zoom function to compensate for the distance and room size. A very quiet motor was tested and chosen to ensure it wouldn't affect voice recognition. It is quite heavy and was also located in the base to assure your Kinect doesn't move around and stays put.

- Microphones [29]: It contains four microphones in the background, one on the left and three on the right. This is why the device is so wide. It was found to be the optimal setup for picking up voices at distance and to help determine where the sound is coming from.

The video and depth sensor cameras have a 640 x 480 pixel resolution and run at 30 FPS. The specifications also suggest that you should allow about six feet (1.8 meters) of play space between you and the Kinect sensor, though this could change depending on where the sensor is standing.

The great advantage of Kinect it that it is able to calculate, with precision of centimeters, the distance of objects and people in the scene within a range from 60cm to 5m in real time, with low-cost hardware, because it uses a USB connector (and an extra power source). It is possible now to adapt it on personal computers, only relying on a driver that works properly.

With all this, many researchers began to apply their skills to develop projects that believed the Kinect would be revolutionary and enthusiastic. Beyond games, this device has created several areas of research and the area of recognition of human gestures to interact with machines is very active, where there are already some projects published. The proposed system will minimize such restrictions, allowing an interaction through gestures to realize

operations in navigation menus.

There are several motion sensors available in the market that can be used as natural user interface for applications and games through the use of gestures and spoken commands. The Kinect sensor by Microsoft is one example. This sensor was and is mainly used for games but the scientific community has been finding many areas where this kind of device can be of major utility.

2.4 Research projects

For many years, researchers have been discarding the more annoying and outdated peripherals and finding new natural interaction systems between humans and machines. From this research area, new technologies appeared like, for example, devices with touch screens, the first mobile phones and some monitors and tablets. Microsoft Surface is a multi-touch product which is developed as a software and hardware combination technology, that allows users to exchange files, and all interactions make modern life simpler by the use of gestures recognition.

When it was launched, Microsoft Kinect was seen as a new way of interacting with games and the world of video games has suffered a major revolution. The latest gadgets have brought a new concept of fun, taking players from the sofa and put them to move, with fun games and focused mainly on sports, dance and health.

If Nintendo Wii has opened a new age, Microsoft Kinect came to make the future much more promising. Microsoft is now taking Kinect motion control system for the Xbox into fields that are beyond those games, because the Microsoft gadget has shown many faces and is now being used for several different purposes.

One of the first new developments with Kinect was the use of an interface without controls for manipulating images. During the first months after the release of Kinect, a developer named Florian Echtler [23] created a program that uses the sensor to resize and treat images. The resizing is done using the same gestures that one would use on a multitouch screen, but this time the user doesn't need any screen at all.

The Kinect contains a series of sensors that translate user's gestures and speech in to actions on the screen. With this set of features, Microsoft expects that these applications

can be used in so different areas like healthcare, education, and high-tech manufacturing.

2.4.1 Revolution in robotics

In this area, many developers want to create more intelligent robots. The company NSK came out with a different and useful idea that takes advantage of this sensor. This Japanese company was able to put a robotic guide dog to work and help the visually impaired. When the robot finds an obstacle, it communicates that to the user, speaking through a metallic female voice.

The movement of this cyber dog is made by small wheels, but that doesn't prevent it from folding legs and climbing stairs. The robot is activated with a single press of a button. [16]

2.4.2 Topography

The use of the Kinect has revolutionized the topography and made it much more interesting. Two students at the University of West Bohemia, Czech Republic, invented the SandyStation a new way of teaching this and other disciplines, they have put together an interactive sandbox using the Microsoft Kinect. It is formed by a gadget from Microsoft(for example), a projector, a computer and a box of sand. The Kinect sensor is mounted above the sandbox to measure the sandbox's topography, and then a projector shines a color-coded overlay onto the sand, showing the depth of each peak and valley. Even water can be mapped. [34]

2.4.3 Archeology

Researchers at a University of California have managed to transform the Kinect into a tool for mapping archaeological places. It was put into operation with a system called Calit2, where the gadget can perform an identification of the three-dimensional terrain. Calit2 has an immersive system called StarCAVE [6], a 360-degree, 16-panel setup, which allows researchers to interact with virtual objects. The system pulls the information captured by the Kinect's camera and infrared scanner, turning the information into avatars that can be plugged into virtual worlds. The hack's creator is hoping that might some day be able to

use the Kinect to scan entire buildings and neighborhoods.

The archaeologists found, in addition to getting the job done more efficiently, they can also save lots of money, since the system device LiDAR[2] was much more expensive. [27]

2.4.4 Caring for older people

Some of the alternative uses of the most praised Kinect are related to health and quality of life. In a Berlin university, scientists have managed to create a wheelchair that can be controlled by one device. The invention is also able to identify and avoid obstacles. [20]

In another institution, the researchers developed a kind of radar to be used in helping people with visual impairments. In September 2011, a project from University of Missouriy tried, to adapt the Kinect in order to be used in monitoring of old patients.

This way, in some American hospitals, the Kinect is being used for identification of diseases in older patients, because the recognition of movements can demonstrate changes, in the way of walking, and thus avoid the risk of falls. [5]

2.4.4.1 Wi-Go

The project that Luis Carlos Inácio de Matos has been working on is called Wi-Go and will definitely help people with disabilities as well as pregnant or old people that go shopping. A mechanism with a basket inside and including a laptop and Kinect will follow the user around the store, so the user do not have to push the basket around the shopping. It follows the user at a safe distance and tracks every move that the user does. [9]

2.4.4.2 Kinect helps rehabilitation therapists

A pilot program at the Royal Berkshire Hospital in Reading UK, shows how Kinect can potentially help individuals who have suffered strokes or other traumas, that have left them with impaired mobility.

Patients in the hospital's neurological rehabilitation unit are using a Kinect-based

[2] **LiDAR** - is an optical remote sensing technology that can measure the distance to, or other properties of a target by illuminating the target with light, often using pulses from a laser.

system that requires them to react to on-screen events and commands. The hospital says the exercises are helping them to regain mobility, coordination, and balance. [22]

2.4.5 Kinect helps kids with Autism

At the Lakeside Center for Autism in Issaquah, Wash., children are integrating all the Kinect technology in their therapy sessions. They laugh, jump or wave their arms, making their on-screen avatars do the same through the use of Kinect motion-sensing technology. The students of University of Michigan designed Kinect games for these children in which they make tickle the avatars that Kinect create on-screen and then they are able to learn about facial expressions and appropriate touch. [36]

2.4.6 The help for surgeries

Among all these features, perhaps the most spectacular is the use of Kinect in delicate procedures. Two ways have been worked out to bring the component of video game console into the operating room.

In procedures that use cameras and electronic scalpels, doctors have used Kinect to have a "real sensitivity" of what is really happening, this is happening, since the typical joysticks used might stop working during a procedure. Students at the University of Washington recently developed a hardware to create 3D maps of the body's interior of the patient and provide force feedback to surgeons. Aside from rumbling, the joystick will actually lock up in certain instances when a surgeon nears an area that could hurt the patient, creating a so called "force field" around specific body parts.

In other cases, with the help of Kinect, the surgeon doesn't have to touch the computer controls to adjust images and view exams. This way, he may save some time and it's necessary to be doing all the sterilization process at each stop. This, also helps reducing bacterial infections transmitted by operating room staff to patients.

This is one of the highlights of this project since it enables gesture recognition and therefore allows the interactive control and navigation of any application with just a simple movement of the arms or hands. [33]

2.4.7 KinEmote

KinEmote is produced and distributed by Lateral flux inc. creators of kinEmote software, that allows windows users to use an easy and free application that takes gestures captured by Microsoft Kinect and translates them into key strokes that any windows application can recognize. The users can easily map their own keyboard keys to each of the gestures recognized by the KinEmote. This means that clicking and dragging becomes a simple gesture of just opening and closing your fist.

The creators of KinEmote released on April 29, 2011 a beta version of their Windows Kinect software that gives full control over a mouse cursor by use of the Kinect sensor. The software tracks a user hand with pin-point accuracy and translates that into mouse movements.

KinEmote allows a user click on something by closing the hand, and drag stuff around by keeping a hand closed. In that time there is no support to do a 'right-click' but software that can be controlled with just the mouse and the left mouse button should be perfectly fine as being controlled by Kinect. With the release of SDK on February 1, 2012 the creators of KinEmote decided to rebuild from the ground up on the MS Drivers and expect to release the new product in March 2012 that will be called "KinEmote MS". [18]

2.5 Literature

This type of project fits in a research area in constant evolution. There are several studies that try to link the technical side of natural interaction with environments in augmented reality, and this is what will be explored in this chapter.

The main purpose of this work is the recognition of gestures, more specifically the recognition of hand for natural interaction. The study of the hand itself is necessary as the first step throughout this hand gesture recognition and interaction research.

For recognition of hands without any object, Lee and Chun [19] present a method for manipulating virtual objects, in a 3D coordinate system. The detection of gestures on virtual objects is done through a system that is designated by YCbCr[3], which is a model of

[3]**YCbCr** - One of two primary color spaces used to represent digital component video (the other is RGB).

color space used to detect the region of hand effectively. In this work, several interactions with virtual objects are supported, including rotation. Despite having good results this work also requires the calibration of the camera, the space must be well illuminated and without complex backgrounds.

Bruno Fernandes [10] introduces a method for interaction in augmented reality, which uses Haar classifiers[4], as a feature for recognition of hand and gestures performed on a desktop. The work environment consists of a plane with the array of ARTag[5] markers, whose task is to delimit the area of operation of the hand, and helps in the detection of collisions. This way, it shows the rotation and resizing of virtual objects in a two-dimensional space, without any significant problems.

In general, all applications have restrictions in the workplace. There are also issues with luminosity and on the skin color because this projects works in a bidimensional space.

In [35] the WagO system, a hand gesture control plug-in for the open source DICOM viewer OsiriX [2], was presented. Hand gestures and 3D hand positions are recognized by using two webcams. However, the system can only recognize a small set of static gestures, the user position is tightly constrained and the recognition rate is not high enough to allow an effective interaction.

Recently, range sensors have stood out as an option to approach human motion capture with a non-invasive system setup. Time-of-flight (ToF)[6] sensors provide, at high frame rates, dense depth measurements at every point in the scene. ToF cameras capture an ordinary RGB image and in addition create a distance map of the scene using the light detection and ranging (LIDAR) [6] detection schema: modulated light is emitted by LEDs or lasers and the depth is estimated by measuring the delay between emitted and reflected light. This approach makes the ToF cameras insensitive to shadows and changes in lighting, so allowing a disambiguation of poses with a similar appearance.

More recently, a less expensive solution to obtain 3D information from video, with respect to the one implemented in the ToF cameras, has emerged: projecting structured IR light patterns on the scene and retrieving depth information from the way structured light

[4]**Haar** - like features are digital image features used in object recognition.

[5]**ARTag** - is an "Augmented Reality" system where virtual objects, games, and animations appear to enter the real world.

[6]**Time of flight** - describes a variety of methods that measure the time that it takes for an object, particle or acoustic, electromagnetic or other wave to travel a distance through a medium.

interferes with the objects in the scene. This is the mechanism used in Microsoft Kinect.

2.6 Development frameworks

There are three major projects that offer freely available libraries that can be used to process sensor data from Kinect: OpenKinect, OpenNI, and CL NUI.

Before the official launch of the Microsoft Kinect sensor, Adafruit Industries announced that they would be giving a monetary reward to anyone who could write software that could allow a computer to access and process data from the sensor. [14] Alex Kipman was able to successfully overcome this challenge two days after the announcement, but it was Hector Martin in October of 2010 who was recognized as the winner since Kipman wasn't prepared and didn't want to reveal his code. [11]

Hector Martin shared his code and released it on November 10 of 2010, which would mark the beginning of the OpenKinect project. [30] This project provides drivers for Kinect, wrappers for different languages and an analysis library. Everything published as open-source. Development of driver SDK and CL NUI Platform was by Alex Kipman's code. The driver and SDK with C / C++ / C# libraries are freely available for use by the public.

The researchers at Microsoft have recently announced the launch of a commercial Kinect for Windows SDK in February of 2012. The initial version promises, among other things, the processing of audio source localization.

All these projects will now be exposed with more detail.

2.6.1 OpenKinect

OpenKinect is a community of people, not a library, whose main outcome is the libfreenect Kinect driver. Libfreenect and OpenNI+SensorKinect are two competing, opensource libraries/drivers. Libfreenect (Apache 2.0 or GPLv2) is derived from a reverse-engineered/ hacked Kinect driver which works across Windows, Linux and MacOS X. OpenNI + SensorKinect is derived from open sourced (LGPL) PrimeSense code. [12]

The OpenKinect project appeared from the interest of using the Microsoft Kinect sensor on PC in November 2010. As previously stated, Hector Martin released the code in the libfreenect Github[7] and on December 17, it was implemented for Win32 platforms, using libusbemu as an emulator of the libusb 1.0 Linux library. From this moment on, the project faced an outstanding evolution. These efforts produced the first open source driver for the Kinect.

The OpenKinect project it is driven by more than 2,000 employees, in which the program focuses on two parts: "libfreenect" and "OpenKinect Analysis Library". Currently, the main focus is libfreenect software. [30]

The libfreenect library has all the code required to activate the communication and data exchange with the Kinect hardware, featuring drivers and a multi-platform Application Programming Interface (API) that runs on Windows, Linux and OS X. The API has bindings and extensions for many languages/platforms like C, C++, .NET, Java, Python and C Synchronous Interface. [12]

On the other hand, the OpenKinect Analysis Library acts as a middleware component, which analyzes raw data and produces useful information's such as tracking, hand tracking, skeleton tracking and 3D reconstruction.

The OpenKinect project doesn't provide detection of gestures out of the box, but this can be accomplished with the use of the OpenCV library, that has image an video I/O(input/output) processing modules, data structures, linear algebra, GUI, mouse and keyboard control, as well as computer vision algorithms (for example, image filters, camera calibration, object recognition, structural analysis and others). [30]

[7] **Github** - is a web-based hosting service for software development projects that use the Git revision control system.

2.6.2 OpenNI

OpenNI or Open Natural Interaction is an organization established in November 2010 and the main objective is to improve the compatibility and interaction with natural interface devices, applications and middleware. OpenNI was formed by a group of companies which includes PrimeSense ltd., which provided the middleware libraries that can convert the sensor data from a compatible device. The libraries are available for applications written in C, C++ and C#, and can be used across multiple platforms.

PrimeSense is responsible for the 3D sensing technology (Light Coding) used in Kinect hardware. Light Coding is the technology that allows building 3D depth maps of a scene in real-time with a high level of detail. The organization is providing an open source framework, featuring an API to write applications using natural interaction, being available for Windows, Mac and Linux and originally written in C++.

The API allows the applications to operate on top of different middleware modules, and with that avoiding the dependency between the sensor and middleware. An important feature consists in the fact that the API provided by OpenNI allows researchers to perform the scanning of three-dimensional scenes of the real world. Through the stream of data, generated from the sensor input, it is possible to represent a body, location of a hand, or even a set of pixels representing a depth map. Also, all these operations can be performed independently of the sensor. [24]

The next figure 2.2 shows the three layers defined in the concept used by OpenNI, having each one the following meaning:

- Application: Represents the software implementations that use gestures and voice interaction;
- OpenNI interfaces: Represents OpenNI core, providing interfaces for communication with sensors and middleware components, which analyze data from sensors;
- Sensors: They represent the hardware devices that capture audio and video elements from the scene.

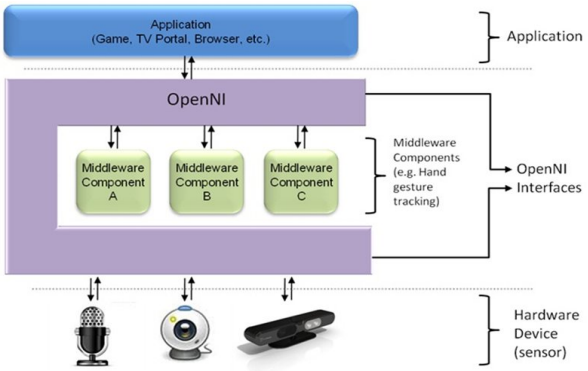

Figure 2.2: OpenNi architecture. [26]

2.6.3 NITE

NITE is a reference project that allows the computer to perceive the 3D world and translate those perceptions into an image of synchronized depth, like humans do. NITE identifies users and tracks their movements, providing the framework API for implementing Natural-Interaction controls on gestures. These gestures allow the users to naturally and intuitively control the living room with nothing in their hands. [17]

The PrimeSense NITE middleware provides the needed infrastructure for application developers to have a free design of their products. NITE is challenging because this middleware offers a comprehensive framework with a rich set of standard and well-documented API that allows users to capture and detect gestures and movements, therefore enabling the tracking of active users and estimating the position in 3D space of important points of the human body namely joints(head, hand, shoulder, elbow and knee). [15]

The NITE engine contains features for gestures detection and recognition, as well as the control of users in the scene who are interacting and exchanging control between them. In the latter case, the gestures made by other users in the scene are ignored, giving the active user control over an uninterrupted experience.

In some cases, if a passive user (not detected) to become active (to be controlled), it is necessary to go through a calibration process: keeping the arms in position (Ψ), like the silhouette of a person in this pose, similar to the shape of the Greek letter “psi”. After a few seconds, the calibration is complete. [1]

Despite supporting multiple users, this middleware is designed to support all types of

action. The NITE middleware, with the help of algorithms for computer vision and image processing, fires user events when someone enters or leaves the scene.

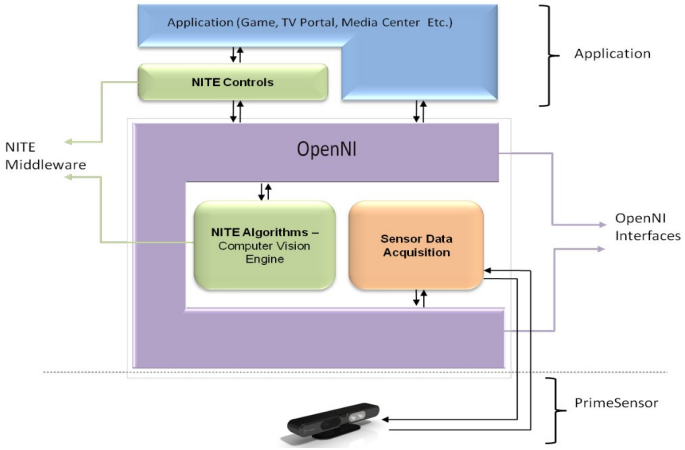

Figure 2.3: NITE middleware. [15]

The developers of the framework ensure the compatibility with new future versions of the PrimeSensor, offering unprecedented performance in terms of rate detection, robustness and efficiency of execution. This middleware can be used with other devices with similar characteristics without the need for big changes in the application implementation. [17]

2.6.4 Windows SDK

On February 21, 2011 Microsoft announced that it would release a non-commercial Kinect software development kit (SDK) for Windows in spring 2011, being later released for Windows 7 on February 1, 2012. The non-commercial SDK beta enables human motion tracking, voice recognition, and depth sensing. The SDK includes drivers and a rich and well documented API, and also good installation manuals and resource materials.

In 1 February 2012, the Kinect for Windows SDK expands the possibilities for innovation with features like Near Mode, which enables the depth camera to see objects as close as 40 centimeters in front of the sensor. In addition, up to 4 Kinect sensors can now be plugged into the same computer.

One of the many improvements of windows SDK is improved skeletal tracking, which

lets developers control which user is being tracked by the sensor. In addition, the latest Microsoft Speech components, along with an improved acoustic model significantly improve speech recognition accuracy.

The API was enhanced with consistency and ease of development, for that new developers should have a much easier time learning how to develop with Kinect for Windows. [3]

In Windows SDK beta version, the Kinect can only be used with Xbox 360 hardware. Applications built with this hardware and software are for non-commercial purposes only. To accommodate existing non-commercial deployments using the SDK beta and the Kinect for Xbox 360 hardware, the beta license is being extended to June 16, 2016. [4]

Chapter 3

Technical aspects of the Kinect sensor

3.1 3D Sensor

The Kinect has become an important 3D sensor and quickly caught the attention of the researchers, thanks to the fast human pose recognition system developed on top of the 3D measurement system.

The Kinect device consists in an IR projector of a pattern and an IR camera, which are used to triangulate points in space. It works as a depth camera, and a color (RGB) camera, which can be used to recognize image content in 3D. As a measuring device, Kinect delivers three outputs: IR image, RGB image, and (inverse) Depth image.

IR Image

IR camera has a resolution of 1280x1024 pixels for a 57x45 degrees FOV, 6.1 mm focal length and 5.2m pixel size. It's used to observe and decode the IR projection pattern to triangulate the 3D scene. If properly illuminated by a halogen lamp and with the IR projector blocked, it can be very reliable.

RGB image

RGB camera has a resolution of 1280x1024 pixels for 63x50 degrees FOV, 2.9 mm focal length and 2.8m pixel size. This camera delivers medium quality images. It can be

calibrated and used to track the camera motion.

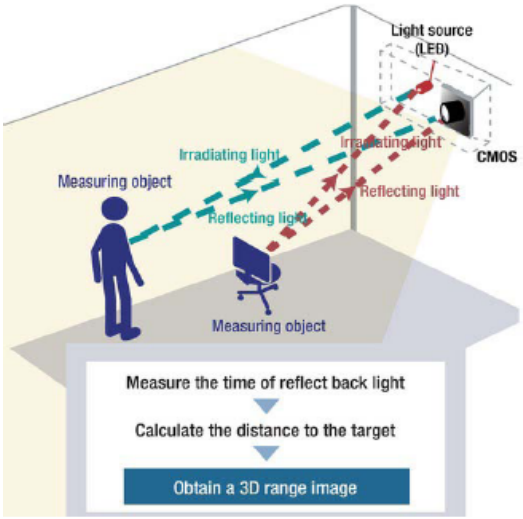

Figure 3.1: How Kinect works.

Depth image

The main raw output of Kinect is an image that corresponds to the depth in the scene. Rather than providing the current depth (z), Kinect returns the "inverse depth" (d). The depth image is built by triangulation from the IR image and the projector and then it is "carried" by the IR image.

The depth camera on Kinect is a pair of laser projector and sensor with a baseline of approximately 7.5cm. It gives us also the per-pixel depth information, but through the use of two properties it is observing the pattern manipulation from projecting a fixed pattern of light and dark speckles to the working environment. The Kinect performs the depth computations itself and drivers provide the raw data streaming from the system. In an ideal 3D camera, the depth is given by the formula,

$$z = \frac{b_f}{d}$$

The variable z is the depth (in meters), b is the horizontal baseline between the two cameras and f is the focal length of the cameras (in pixels). d is the disparity,

$$\mathrm{d} = \mathrm{x}_L - \mathrm{x}_R$$

x_L and x_R are the position of a same point in the space on x axis of the image plane. On Kinect, the depth is acquired differently from the normal 3D camera,

$$z_{perpixel} = \frac{b_f}{\frac{1}{8}(d_{off} - k_d)}$$

d_{off}[33] is about 1091.5, b is about 7.5 cm, k_d is a disparity map built-in, 1/8 means k_d disparity map is in 1/8 pixel units and f is about 580 pixel units.

The raw depth image needs to calibrate on the RGB camera to make the pixel pairs match on both colour and depth image, and this calibration is built-in in a framework, like OpenNI. The pixel value on the depth image is a 16bit integer range in 2^{11}. [13]

For example, knowing the depth value of the global position of hand, the depth segmentation is simply done by acquiring pixels within a certain depth range around the detected position. The hand depth image is sliced according to the global position at the same size, so that the global position of the hand in each sliced hand depth image will not change.

3.2 Protocol documentation

The interaction between components is represented in the below Figure 3.2. Each component sends a specific action through an event so the Kinect can perform certain tasks between them. For example, the Kinect sensor sends an input data when it detects a gesture and then sends an event to the appropriate component.

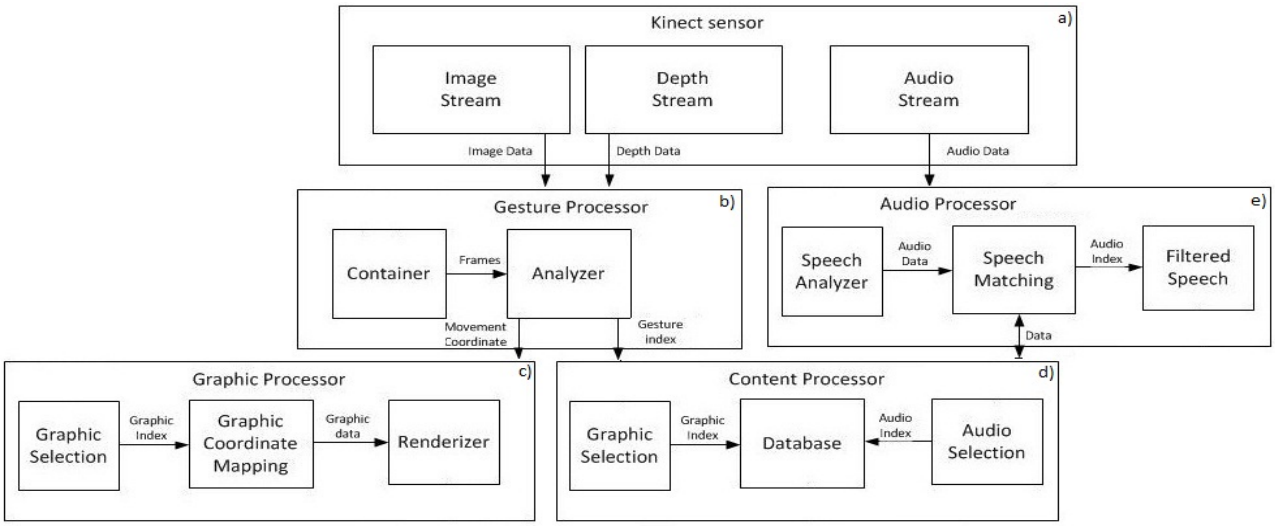

Figure 3.2: Components of the Kinect.

a) Kinect sensor- input of user's movement and speech; b) gesture processor- process of movement and carry out task and mimic the user; c) graphic processor- handle and render graphics; d) content processor- process individual contents module; e) audio processor- process of speech and output a filtered tone of the user.

The Microsoft Kinect has three devices that are part of the communication with the computer. As mentioned before, the Kinect sensor has natural user interfaces(NUI), namely the NUI motor, NUI cameras(RGB and Depth) and NUI audio.

The Kinect features an USB hub, being an easy way for the device to combine three separate chips using a single USB connection.

In a control packet structure exists a function (USB control of messages) that sends a simple control message for a specified endpoint and waits for the message to complete, or timeout. This function is used to read accelerometer values, internal motor, led and camera registers. Each request starts with an 8 byte long setup packet which has the following format:

Offset	Field	Size	Value	Description
0	bmRequestType	1	Bit-Map	**D7 Data Phase Transfer Direction** 0 = Host to Device 1 = Device to Host **D6..5 Type** 0 = Standard 1 = Class 2 = Vendor 3 = Reserved **D4..0 Recipient** 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4..31 = Reserved
1	bRequest	1	Value	Request
2	wValue	2	Value	Value
4	wIndex	2	Index or Offset	Index
6	wLength	2	Count	Number of bytes to transfer if there is a data phase

Table 3.1: USB control messages

3.2.1 NUI Motor

The Kinect motor communicates through a single control endpoint, which means that all the messaging between the code and the device can be carried out in terms of control transfers. Endpoints are a type of USB "data pipe" and there are 4 kinds:

- **Bulk** endpoints are for transferring a lot of data, like a disk drive. It takes a little longer because transfers large packets. This endpoint goes only in one direction.

- **Interrupt** endpoints are for transferring tiny amounts of data very quickly, like for a USB mouse. In this case, the device has to be responsive because we want fast movement. This endpoint goes only in one direction.

- **Isochronous** endpoints are for transferring a fair amount of data where the data must show up at the same time and if it can't it should just be dropped. This is for audio and video where the time is fundamental. This endpoint goes only in one direction.

- **Control Endpoints** They are used to transfer small amounts of data to turn a device on or off.

Every type of USB device must have a unique VID (idVendor) and PID (idProduct). The VID is the manufacturer. In this case, 0x045e is the VID for Microsoft. All Microsoft products will have that VID. Each product has a different PID, so all Kinect Motors use PID 0x02b0 and this doesn't differ between two Kinects, they both have the same PID. The VID and PID are used as a way for the software driver to find the product.

The descriptor of the device is determined thereafter. A descriptor is a sort of "menu" of what the device can do and how transfers data. This motor device has no endpoints, but that doesn't mean that there's no communication with it. It just means it only uses a bidirectional control endpoint, and this isn't surprising because motors are slow and don't require a lot of data to control.

The motor device controls the current motor for tilting/panning the Kinect. After the driver for the motor is installed/activated, Kinect displays a flashing green light and the other devices connect to the internal USB hub. The tilting of the camera, in order to get the proposed angle, is relative to the horizon, and not relative to the base.

The accelerometers in the Kinect are used to detect when it has reached the correct angle, stopping the motor afterwards. The value is in range -128 to +128, where -128 is a bit less than -90 degrees, and +128 is a bit less than +90 degrees.

The joint range of the Kinect motor is +31 degrees up and -31 degrees down. Overcoming this range can stall the motor. There isn't a way of knowing the angle in relation to the base because the angles are measured in relation to the horizon.

The last byte of the accelerometer report show the motor status in which 0 means stationary, 1 means stopped, etc. The second byte (out of 10) is the level of "effort" on the motor, where can supervise the "applications", which 0 means no effort.
The accelerometer basically consists of a sensor element and an ASIC[8] packaged in a 3x3x0.9mm LGA[9] Dual.

An ASIC, using a standard CMOS manufacturing process, detects and transforms

[8] **ASIC** - is a microchip designed for a special application, such as a particular kind of transmission protocol or a hand-held computer.

[9] **LGA** - is a type of surface-mount packaging for integrated circuits because for having the pins on the socket rather than the integrated circuit.

changes in capacitance into an analog output voltage, which is proportional to acceleration. The reading accelerometer data is stored in two byte pairs for x, y and z. The motor of the Kinect is controlled byset of control messages, some of them being bmRequestType, wValue, etc. The LED device is available in several colors like green, red, etc.

3.2.2 NUI Cameras

The most interesting device is the camera that provides both an RGB and a depth map image. This device has two isochronous endpoints, being both IN type (data going IN to the computer). One of these provides the RGB image data and the other provides depth map data. In addition to the default control endpoint, the camera device has:

- Endpoint 0x81 - IN, Isochronous, 1920 bytes (color camera);
- Endpoint 0x82 - IN, Isochronous, 1760 bytes (depth camera).

The device initialization of the cameras happens through control messages. The initialization headers and replies are expected to be a packed struct. The "magic" value for initialization commands to the camera matches the ASCII[10] string "GM", for initialization replies from the camera matches the ASCII string "RB".

The data is always 4 bytes, but values are different from packet to packet. The frame transfer happens through isochronous transfers from the two device endpoints.

3.2.2.1 Depth Camera

Most of the gesture control systems were based on the ToF method.

ToF cameras obtain the depth information by emitting near-infrared light which is reected in the 3D surfaces of the scenario and back to the sensor. (see Fig. 3.1). Currently two main approaches are being employed in ToF technology [8].

The first one consists in sensors which measure the time of a light pulse trip to calculate depth. The second approach measures phase differences between the emitted and received signals.

[10] **ASCII** - codes represent text in computers, communications equipment, and other devices that use text.

Many ranging / depth cameras have already been developed. The Mesa Imaging SwissRanger 4000 (SR4000) is probably the most well-known ToF depth camera. It has a range of 5-8 meters, 176 x 144 pixel resolution over 43.6 x 34.6 FOV and operates at up to 50 FPS. [21]

The PMD Technologies CamCube 2.0 is lesser-known, but an equally impressive ToF depth camera. It has a range of 7 meters, 204 x 204 pixel resolutions with 40.0 x 40.0 field of view and operates at 25 FPS. [28]

The majority of gestural control systems is based on the ToF method, and consists in sending an infrared light, or similar, into the environment.

The time and wavelengths of light that returned to the capture sensor inside the camera will then give information on how the environment looks like. The Kinect instead uses a technique of encoding the information in light patterns that are sent out. The deformations of those patterns will be similarly captured and they will return information on the environment as it was for the other method.

When the camera receives the IR light back, the real processing can start. Prime-Sense developed a chip that is located inside the camera itself, which is already able to process the image by looking for shapes resembling the one of a person. It tries to detect the head, torso, legs and arms. Once it has achieved this, it starts computing a series of parameters like where the arms are probably going to move.

The depth camera analyzes IR patterns to build a 3D map of the room and all objects and people within it, and sends a monochromatic image made up of 11-bits or 2^{11}=2048 values per pixel, in big-endian[11] format. These values can be shifted and repeated into an RGB888 (RGB, uses 8 bits each for the red, green, and blue channels) pixel format to make a gray scale image based on the depth.

3.2.2.2 RGB Camera

The RGB camera follows the same format as the depth camera. The frames for the RGB camera are received isochronous endpoint. The RGB Camera lays out frames in a 640x480

[11] **big-endian** - this machine stores the most significant byte first.

Bayer pattern[12] wich is RG, GB.

3.2.3 NUI Audio

The audio device has several endpoints that provide enough bandwidth for a perfect audio both in and out of the device. In addition to the default control endpoint, it also features other endpoints in two kinds, bulk and isochronous, both as IN and OUT type. That IN and OUT are relative to the USB host, and OUT means sending data from the computer to Kinect.

[12] **Bayer pattern** - is a color filter array for arranging RGB color filters on a square grid of photo sensors. The filter pattern is 50% green, 25% red and 25% blue.

Chapter 4

Kinteract High Level Design

In this chapter, an overview of the developed driver/daemon, named Kinteract, is given, providing some detailed information about the modules and components that are part of this application or that interact with it anyhow.

Kinteract is a driver that was developed having in mind using gesture recognition to control any application. It was developed under the Linux operating system, using the OpenNI / NITE frameworks. Another Kinteract feature is that it is ready to be applied to any kind of system that requires menu navigation on the system, because our project is well structured to be a generic driver for navigation menu.

After a preliminary study, we have decided to use the OpenNI/NITE frameworks has a initialization process, to help in the development of this project. OpenNI plays an essential role in easing system initialization and communication with the sensor, while NITE helps in gesture recognition.

Another module that should be mentioned and required for this project is hardware driver provided by PrimeSense. This module makes the recognition of all the Kinect sensor features (hardware) such as RGB camera, IR camera, and USB interface.

4.1 Component overview

There are several components that are used by Kinteract to achieve its goals. Apart from the Kinect sensor, Kinteract uses OpenNI and NITE frameworks to process sensor data and detect gestures. The OpenNI / NITE frameworks were chosen for data processing, more specifically the NITE framework acting as a bridge between the sensor and Kinteract.

The uinput module of Linux kernel allows handling the input subsystem from user land, and it can be used to create and to handle input devices from an application. The operation of this module serves only to generate operating system events, more specifically keyboard events. Because of this, Kinteract has a configuration file that indicates to the application which event will be sent to the uinput layer for each detected gesture.

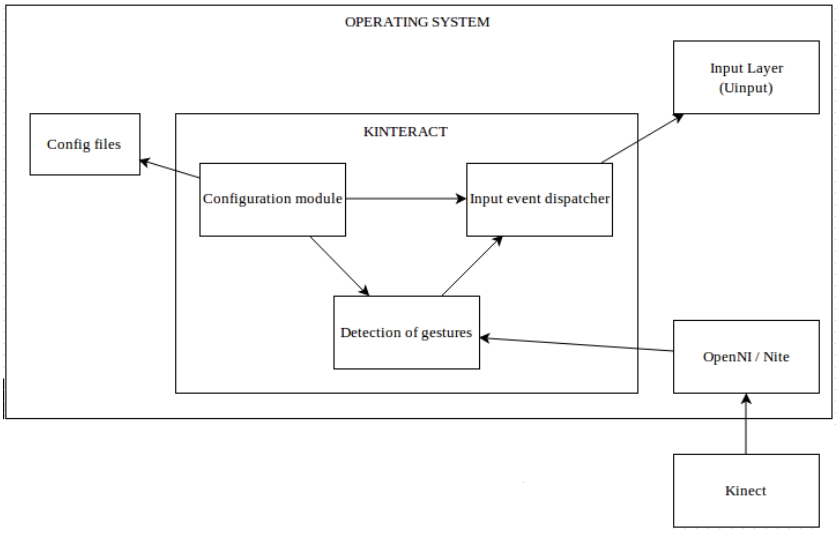

Figure 4.1: Kinteract internal modules.

4.2 Kinteract design

The Kinteract has a initialization process for user registration, in other words when we do any hand gesture it will make the detection of the hand, and so this movement will act as a "Login" or a registration that the user must perform. To better understand how Kinteract works, it is here shown the main modules that form the application and how they interact with each other:

- configuration module;
- detection of gestures;
- input event dispatcher.

In this first point we will show the first steps of the configuration for framework OpenNi inside Kinteract. Initially, OpenNi needs to be configured to work correctly, and after this it will interact with the state machine that handles transitions between states caused by the initialization process, gesture recognition events and possible failures.

Most of the states will be related to the session while it is in progress like, for example, waiting for a gesture, losing track of the hand and re-gaining it or losing connection to the sensor (if, for instance, the sensor is disconnected), in which case Kinteract will be waiting for a new connection without stopping the execution.

The detection module can recognize six gestures ("Up", "Down", "Right", "Left", "Ok", "Back") with the help of the NITE framework, that has important modules for this type of gestures. The module sends the events to the system and uses the uinput module of the Linux operating system. For the user to see the interaction of the gesture on the system, the detection of gestures will look for information about each event, which is on configuration file of Kinteract.

In short, after reading its configuration, Kinteract will do the necessary initialization followed by a cycle in which it will wait for gestures and will proceed accordingly, based on the provided configuration, sending a keyboard event to the Linux input layer. This model allows Kinteract to have a generic use in any Linux system.

The Figure 4.2 is a state diagram that better reflects Kinteract's behavior and internal logic.

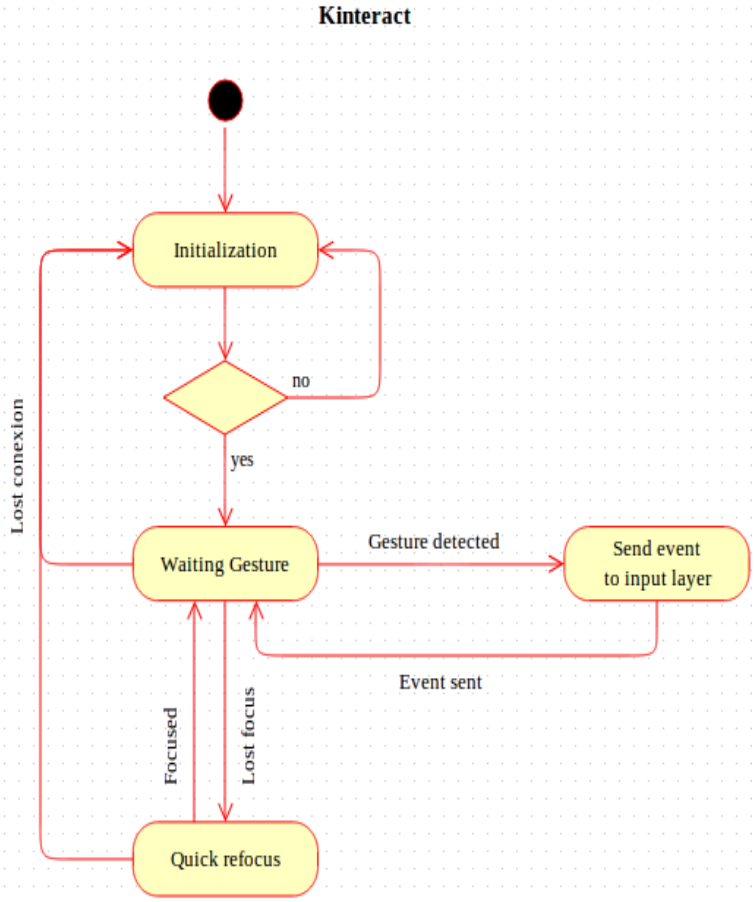

Figure 4.2: Kinteract behavior.

The behavior of Kinteract when connected to a system is carried out as follows. First, it enters in an initialization state, followed by a state that waits for the registration gesture, the "login". After that, Kinteract goes to a state where it starts detecting gestures that were previously registered in the initialization state. For each recognized gesture, Kinteract processes it and sends an event into the input layer of the operating system. If the Kinect sensor loses the user's hand, Kinteract enters a state for quick refocus in order to quickly detect the hand again, returning then to the state where it waits for gestures.

If Kinect sensor loses connection with system during the session, Kinteract handles that failure properly and immediately enters the first state, where it periodically tries to do the necessary initialization of the sensor, resuming normal operation if the sensor is detected again.

For requirement analysis we want Kinteract provoke events with the gestures already

defined, and put those correctly function in a system already made. For be a general driver, we created a XML data for change the input events.

Chapter 5

Implementation Details

In this chapter there will be a deeper explanation of some implementation details related to Kinteract. There will be a strong focus on the technical side regarding each of the used and implemented modules, revealing their importance and how they interact with each other.

5.1 Kinteract details

To use the OpenNi framework it is necessary to initialize some basic methods, such as the *Context* module, which can be the starting point. First we need to construct a context object and initialize it, because the context class has the necessary information to initialization like, *init()*, *InitFromXmlFile ()*.

The first one is for building the context's general software environment. The context object is one of the most important objects in the framework OpenNI, because it holds the state of application and all the other applications like nodes. We use for initialization the *InitFromXmlFile ()*, because it's the combination of two methods that the context object has (*Init ()* and *RunXmlScriptFromFile ()*) that serves to initialize the context object and then create a production graph [25]:

```
1 #define SAMPLE_XML_PATH "Sample-Tracking.xml"
2 rc = g_Context.InitFromXmlFile(SAMPLE_XML_PATH, g_ScriptNode,&errors)
```

The *SAMPLE_XML_PATH* is the file path to the XML file that framework OpenNI brings, and inside has the license and key of instalization and the resolution of kinect sensor mostly. The variable *g_ScriptNode* is a ScriptNode object as the root to all the nodes created from the XML file. The errors is a EnumerationErrors object that show the errors. When initialized, we can create production nodes with *StartGeneratingAll()* method:

```
1 g_Context.StartGeneratingAll();
```

After this, we call the *WaitAndUpdateAll()* method, that waits for all the nodes to generate new data and then updates them:

```
1 g_Context.WaitAndUpdateAll();
```

Before calling this last method we need to initialize the NITE middleware and all the necessary methods for Kinteract to detect all the gestures. So far the program starts by setting up a range of OpenNi nodes and next we will initialize the NITE middleware to detect the gestures. Then it enters in a state machine which passes Kinect events arriving at OpenNi's context object to NITE's SessionManager. To initialize NITE, we call the *new XnVSessionManager* method that creates an object to supply a stream of hand points:

```
1 g_pSessionManager = new XnVSessionManager;
```

The SessionManager's input arguments specify the focus and refocus gestures: a "Click or Wave" usually initiates a register session, but in this project we use a simple gesture like "RaiseHand" to make session registration, and it is also used to resume a stalled session:

```
1 rc = g_pSessionManager ->Initialize(&g_Context, "RaiseHand", "RaiseHand");
```

The NITE SessionManager object creates six detectors for processing the hand points output. In this project we use two of the detectors that SessionManager provides: SwipeDetector and the PushDetector. The SwipeDetector detects only the motion of the hand, with

movements like "Up", "Down", "Left" and "Right". The PushDetector recognizes a hand movement as pushes towards and away from the sensor, like "Ok" gesture.

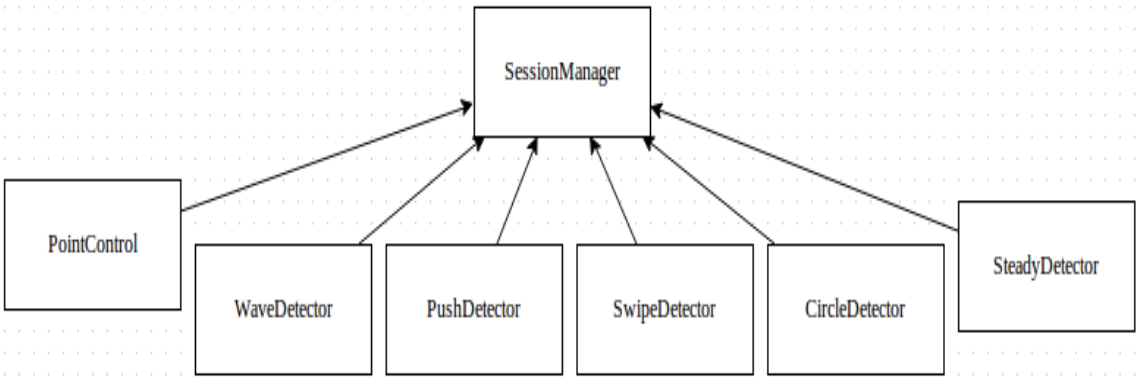

Figure 5.1: SessionManager objects.

In the main module we define three callbacks, one for the beginning of the session, one for the end of the session, and finally another to define the gesture "Back":

```
1 g_pSessionManager ->RegisterSession(NULL,&SessionStart,&SessionEnd);
2 g_pMainSlider ->RegisterItemSelect(NULL,&MainSlider_ItemSelect);
```

The SessionStart is a callback only to start the session and the callback SessionEnd indicates the end of the session. The MainSlider is a SelectableSlider2D object of framework NITE and the callback *MainSlider_ItemSelect* is where the "Back" gesture was defined. To register these three callbacks we used the RegisterSession of the SessionManager and to register the "Back" gesture the RegisterItemSelect of the SelectableSlider2D, both modules of the NITE framework. The rest of the gestures are defined in another class, which contains the callbacks for all the rest of the gestures and uses the SwipeDetector and PushDetector object to define them. However, these same gestures are registered and also added in the SessionManager on the main file, for the user's hand to be tracked and all the defined gestures to be recognized in Kinteract:

```
1 g_Swipes[0]->RegisterCallbacks();
2 g_pSessionManager->AddListener(g_Swipes[0]);
```

In "Back" callback we use NITE's SelectablesSlider2D module because it can control the X and Y axes and also controls the movement along the Z axis. Therefore we can use this module to detect the "Back" movement. It should be noted that this module is also capable to detect and control the movements "Up", "Down", "Left", "Right" and "Ok" but we decided to use SwipeDetector and PushDetector because both allow controlling the minimum duration of the gesture, the angle of the gesture and other things that we find interesting for this project.

5.1.1 Configuration

Kinteract aims at being completely configurable. Libxml is a C language library implementing functions for reading, creating and manipulating XML data. The use of this library was essential for Kinteract to be more practical and was necessary to set some values and data that are related to gesture recognition and input event.

This way, Kinteract can be adapted to any system simply by setting up its configuration XML file. In this file we can defined the necessary fields to control the velocity of gestures(line 1) like "Left", "Right", "Up", "Down", "Ok" and "Back" and also be able to change which events will be fired to the Linux input layer. Next we will find two examples contained in the configuration file of Kinteract:

```
1 <PushVelocity> 0.7 </PushVelocity>
2 <event_LEFT> KEY_LEFT <event_LEFT>
```

The goal of the first line is control the velocity of "Ok" gesture that will only be detected from that velocity, the next line serves to generate the "Left" event (*KEY_LEFT*) on the left gesture. We used the libxml library for the configuration file and load the XML file as follow:

```
1 xmlDocPtr doc;
2 xmlNodePtr node;
3 doc = xmlParseFile("config.xml");
```

We declare *xmlDocPtr*, that is a pointer to the structure of the tree and the *xmlNodePtr* is a pointer to the structure, and is used in traversing the document tree, in other words a node pointer. The *xmlParseFile* determines whether the file is well formed or complies with a specified grammar. Next, to check if we were loading the file correctly, we retrieve the root element of the file.

```
1 node = xmlDocGetRootElement("//kinteract");
```

Then we need to make sure the document is in the right type and we verify if the root element of the document is Kinteract.

```
1 if (xmlStrcmp(node ->name, (const xmlChar*) "kinteract")) {
2   fprintf(stderr,"document of the wrong type, root node != kinteract");
3   xmlFreeDoc(doc);
4   return;}
```

Now we go through the nodes of the tree and in this case we are looking for all the elements that are children of “kinteract” and the parsestory function will find the element that we need.

```
1 node = node->xmlChildrenNode;
2 while (node != NULL) {
3   if ((!xmlStrcmp(node->name, (const xmlChar*)"kinteract"))){
4     parseKinteract(doc, node);}
5 node = node->next;}
```

The *parseKinteract* contains the search of all the elements of gestures velocity. Another function is *parse_Events*, which verifies what event the configuration file has and then load into the right place of the source code.

5.1.2 Uinput

The module uinput can be used to create and to handle input devices from an application. It works, for example, when we have a keyboard and press a key, this key is converted into a code called scancode. Each key has a scancode when is pressed and when it's released. From here, there is a driver that is responsible for reading these scancodes and converts them into other codes called keycodes. The keycode is a code representing a key within a Linux system.

One of the objectives of Kinteract is to be a tool that could transform the gestures in input events and inject these events in the Linux input layer, to be able to control any application. The input layer is the part of the Linux Kernel that manages input devices like keyboard or mouse. Then the kernel exposes the user input layer through a range of defined APIs.

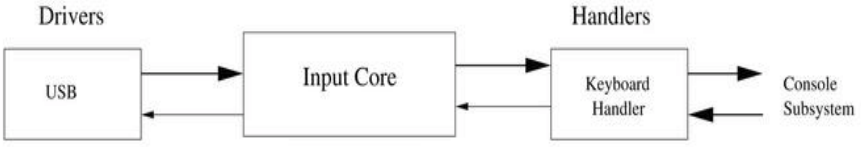

Figure 5.2: Relationship between the Elements of the Input Subsystem.

The three important elements in an input subsystem are the input core, drivers and event handlers, and for example in this case, if the type is a key or button, the code tells which key or button it is, and value tells if the button has been pressed or released. First we have to check the module is installed because this device represents the interface between the application and the kernel input subsystem. Then we need to open the character device in write-only and non-blocking mode, and to finish the configuration by using the *struct uinput_user_dev* from linux/uinput:

```
1 #define UINPUT_DEVICE_NODE "/dev/uinput"
2 struct uinput_user_dev uinp;
3 struct input_event event;
4 fd = open(UINPUT_DEVICE_NODE, O_WRONLY | O_NDELAY);
```

Next we need to configure our input device, and inform the input subsystem which types of input events we want to use. For that we need to verify the type of the events that we need. Then we used *EV_KEY* that represents the key press and release events and we need to describe which keycodes are sent via the input subsystem:

```
1 ioctl(fd, UI_SET_KEYBIT, KEY_UP);
2 ioctl(fd, UI_SET_KEYBIT, KEY_RIGHT);
3 ioctl(fd, UI_SET_KEYBIT, KEY_DOWN);
4 ioctl(fd, UI_SET_KEYBIT, KEY_LEFT);
5 ioctl(fd, UI_SET_KEYBIT, KEY_ENTER);
6 ioctl(fd, UI_SET_KEYBIT, KEY_ESC);
```

The lasts steps are writing this structure to the uinput file descriptor and the creation of the device. The *UI_DEV_CREATE* registers a new device with the input subsystem.

```
1 if (write(fd, &uinp, sizeof(uinp)) < 0) { }
2 if (ioctl(fd, UI_DEV_CREATE) != 0) {
3   perror("set evbit: UI_DEV_CREATE");
4   return -1;}
```

Finally comes the process of sending an event to the input layer. The input event has 3 important fields:

type $\longrightarrow$ the event type, *EV_KEY* in this case;

code $\longrightarrow$ could be either a key code when we are using *EV_KEY*;

value $\longrightarrow$ for *EV_KEY* may be 1 (press) or 0 (release).

The *press_button*() function describes how a keyboard input event is sent to the input subsystem. This function injects a key press event in the input subsystem.

```
1 void press_button(int x){
2   memset(&event, 0, sizeof(event));
3   gettimeofday(&event.time, NULL);
4   event.type = EV_KEY;
5   event.code = x;
6   event.value = 1;
7   if (write(fd, &event, sizeof(event)) < 0)}
```

5.2 Class diagram

To end this chapter we draw a diagram of classes, revealing their structure and main functionalities, and how they interact with each other.

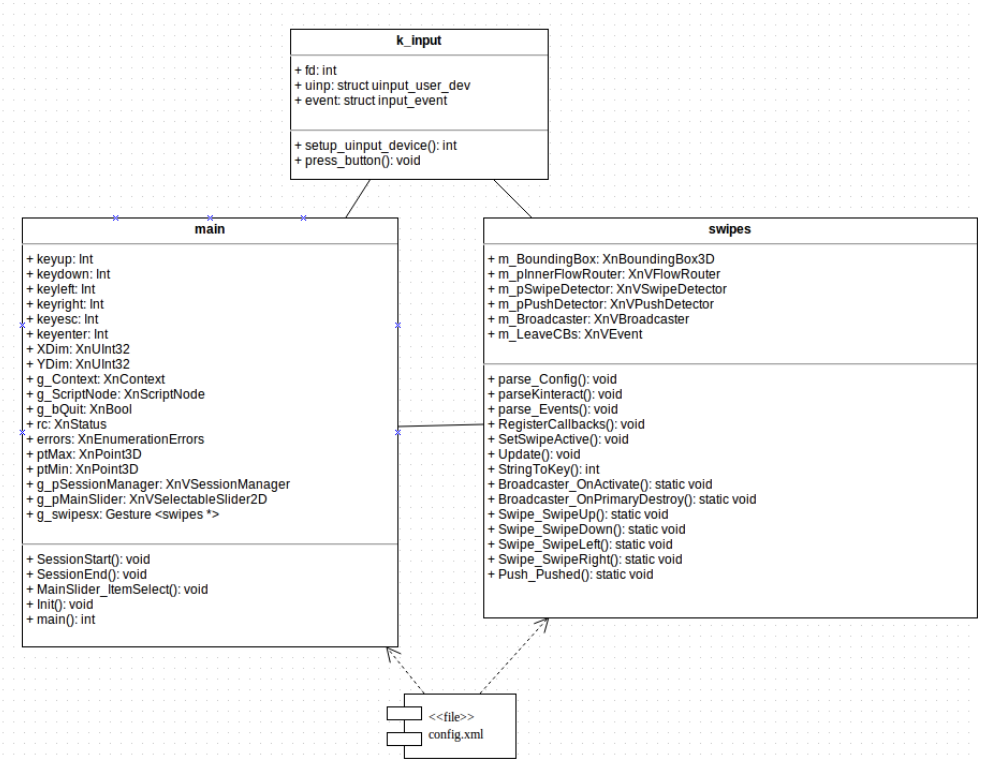

Figure 5.3: Class diagram of Kinteract.

In Figure 5.3 we have in the main file the reference to OpenNi and NITE modules, and the three callbacks for the start and end of the session, which are used when we initialize the SessionManager of NITE. The *MainSlider_ItemSelect()* function is the callback for the "Back" gesture.

The main file picks up everything that belongs to the class swipes, that was defined in the swipes file, because in this class are defined five gestures and, as described above, we just create an object of the type Gesture for Kinteract to catch these gestures correctly:

```
1 Gesture* g_Swipes[1];
2 g_Swipes[0] = new Gesture(ptMax, ptMin);
```

In the swipes file we have the Gesture in class which are defined the gestures "Left", "Right", "Up", "Down", "Ok". The first four are gestures of SwipeDetector type and the "Ok" gesture of PushDetector type. The FlowRouter is a message listener that holds another single listener internally, sending any message it gets to that listener. The Broadcast object sends all the data (hand points) it receives to all the objects that are connected to it. These two files (main and swipes) at the beginning of the session will also interact with the configuration module. The elements of the configuration will load data such as velocities and the events that gestures will generate.

Finally there is an interaction between the k_input file with main and swipes files, where all the defined gestures have a call to the *press_button* function. This function is located in the k_input file, and is responsible for the process of sending an event for the uinput kernel module. In main file there is also the definition of "Back" gesture and so there is a communication with the k_input to process the event related to that gesture.

Chapter 6

Results

In this chapter we present the results of the tests made with five people that shared a perfect profile(old people).

The main purpose of these tests was to get a reaction of individuals to whom this system is directed the old people. We needed to know what kind of gestures Kinteract should have defined to be simple to use by this people.

When we started to design and develop Kinteract, we defined six gestures that we thought could be more intuitive for old people, but there was the need to validate if they would be a good selection and if older people would be comfortable with this kind of gestures.

We want to identify gestures that are naturally performed by subjects when they intend to give commands to the system. These gestures are produced by the subjects without any directions from the interface developers. Finally we want measure the degree of success of the produced gestures in the achievement of the intended interface events.

For that, five people, ageing between 60 and 73 years, without any physical limitation in the arms and hands, were invited to perform some gestures when faced with a real system. This way, we could later check if Kinteract should be adapted for persons in this age range.

6.1 Methodology

At the start of each test, we proceeded with a brief explanation of the system with which they would be faced and clearly reveal the objectives of the tests. Three of the five people already knew the system because they already had direct contact in other tests in this company. The other two, we had to explain in more detail all the process of the tests.

The system used was from the eCAALYX project, an European Research project which is part of the AAL Joint Program, that aims to create a holistic solution that improves the quality of life of older adults with chronic conditions, by fostering their autonomy, while, at the same time, reducing hospitalization costs.

As expected, the interface is very accessible and simple, so the users can navigate with a simple remote, and have access to his latest vital sign measurements, a medical agenda, videoconference with doctors, etc, like shown in Figure 6.1 and Figure 6.2.

Figure 6.1: Menu of eCAALYX

Figure 6.2: Interface of the user weight

One of the major goals of this work is to appeal to movement and mobility in older people. With this set of tests, we wanted these particular persons to help and elucidate us about the gestures that would be more intuitive to interact with a control/navigation menu.

To not get biased results, never at any time we have explained the Kinteract system and the objective of the tests, so the subject could not be influenced by the current status of the system and the currently detected gestures. After having all points clear, we performed

the test by following these steps:

1. First we performed a small test, to check if the person clearly understood the scope of the tests;

2. Second, we began recording them, and we asked the subject to carry out the gestures that he would feel to be more intuitive to perform a determined set of events;

3. Third, we asked the subject to really interact with Kinteract (explaining what were the defined gestures at that time), so we can get a feeling on how could be the reaction of using Kinteract in people with that age.

In the first step, we explained in practical terms the system, showing that with a simple remote control, with which old people are usually familiar, we could navigate the menus with the keys "Left", "Right", "Up", "Down", "Ok" and "Back". After they interact and navigate the system with the remote control, and having already a perception of what would be requested in the next steps, we started to collect some gestures.

In the second step, and one of the most important, we asked them to perform the gestures one by one, always having them thinking on which gesture would be more intuitive to navigate in the menu system. In some cases it was difficult for them to imagine the gestures, so with the help of the remote we tried to explain through buttons like for example, "if we wanted to go left I would press the left button, and now without the remote what would be the best movement that would do to perform the same event?"

In the third and final step, we were able to show Kinteract with the gestures that were already implemented. The main reason for this was to have a perception of how they would adapt to using a system with these characteristics, and also to see the reaction of using the natural interaction in these people with the whole project.

6.2 Identifying intuitive gestures

This section is related to the second step of the test and we will try to demonstrate, in images, the gestures that each senior did for each determined movement.

In the first case we have a male (user 1) with 62 years old who has done the following gestures:

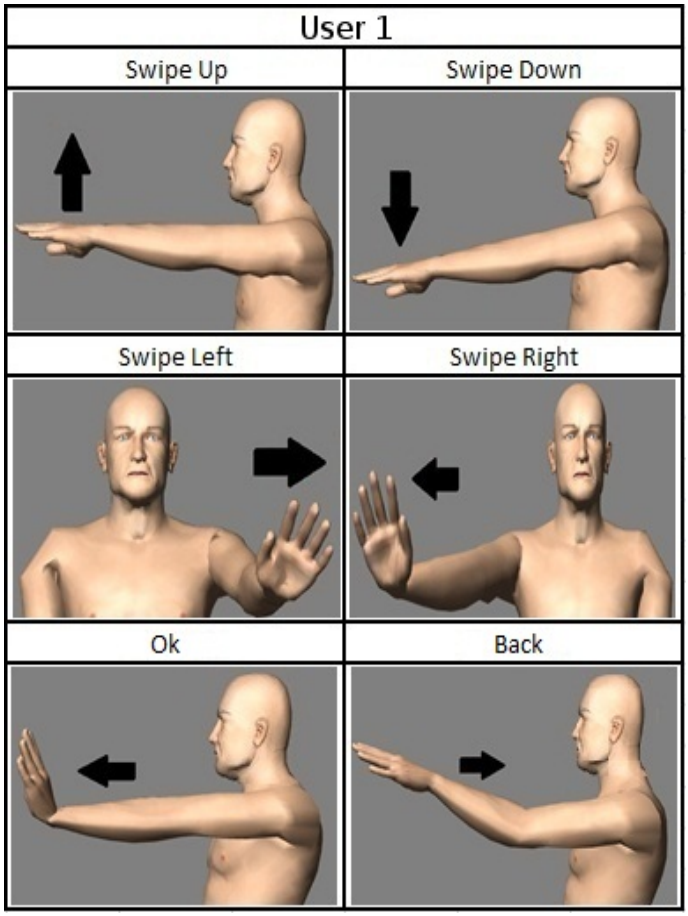

Figure 6.3: Gestures performed by user 1

A note here for the "Up" movement, which this elderly performs with his hand and his fingers in the direction of motion. Another important feature of this test was that, in the case of "Left" and "Right" gestures, this elderly used the two hands, one for each of the movements, despite having been warned to perform with only one hand.

In the second case we have a female (user 2) with 68 years old who has done the following gestures:

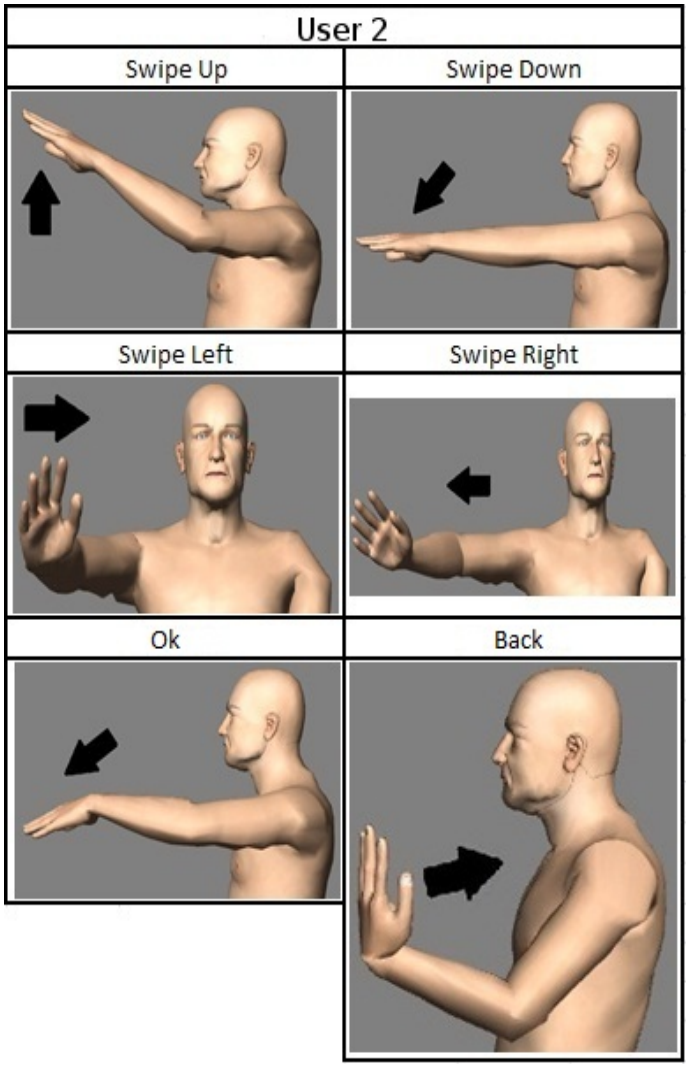

Figure 6.4: Gestures performed by user 2

In this case, we have to highlight the gestures "Down" and "Ok" because she defines them in same manner, practically identical. Another important point of this test is that she had many doubts to find the most intuitive gesture to do the "Back" movement.

In the third case we have a female (user 3) with 60 years old who has done the following gestures:

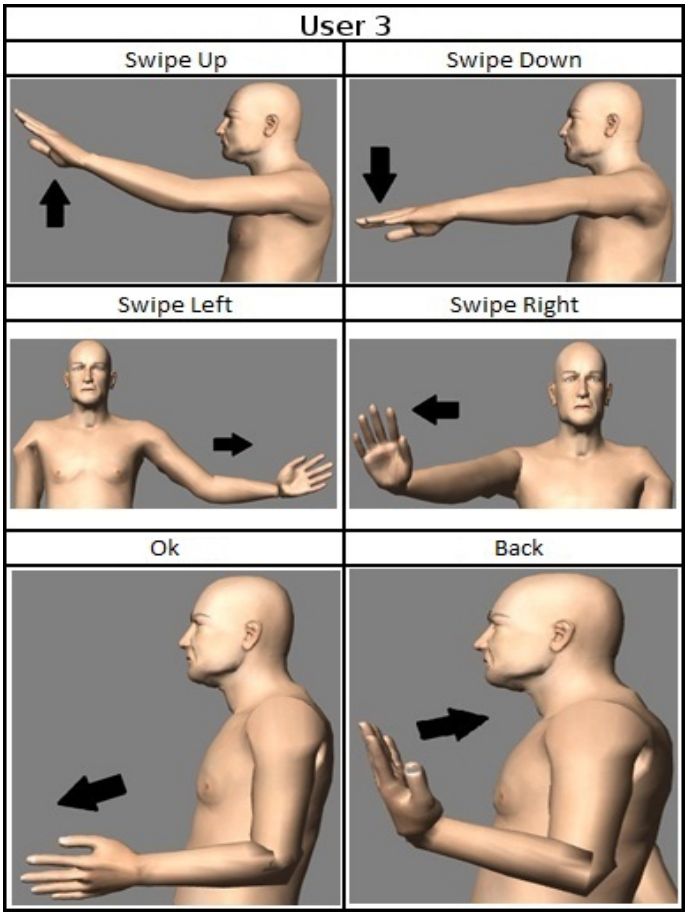

Figure 6.5: Gestures performed by user 3

It should be noticed that, in the "Ok" movement, this person had some doubts to find the simplest gesture. The image isn't very revealing, but the gesture is like pressing something with the hand forward. Another issue is in the "Left" and "Right" movements, because also here the hands have been alternate. On the "Back" movement, the first reaction was also to use both hands.

In the fourth case we have a male (user 4) with 73 years old who has done the following gestures:

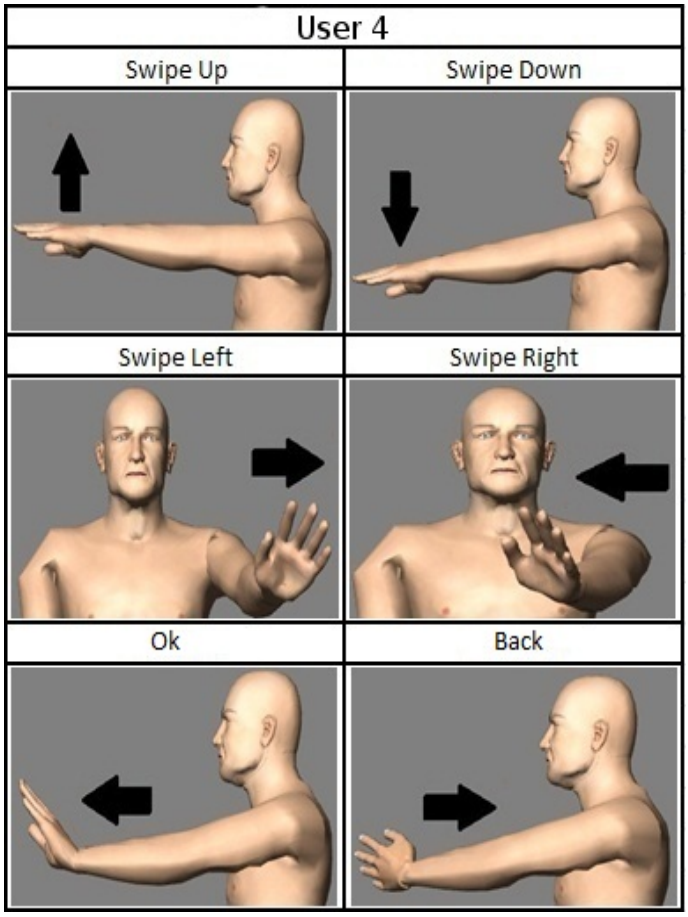

Figure 6.6: Gestures performed by user 4

The only thing to point out in this test is regarding the "Ok" movement, because this elder found it confusing to define this movement.

In the last case we have a male (user 5) with 64 years old who has done the following gestures:

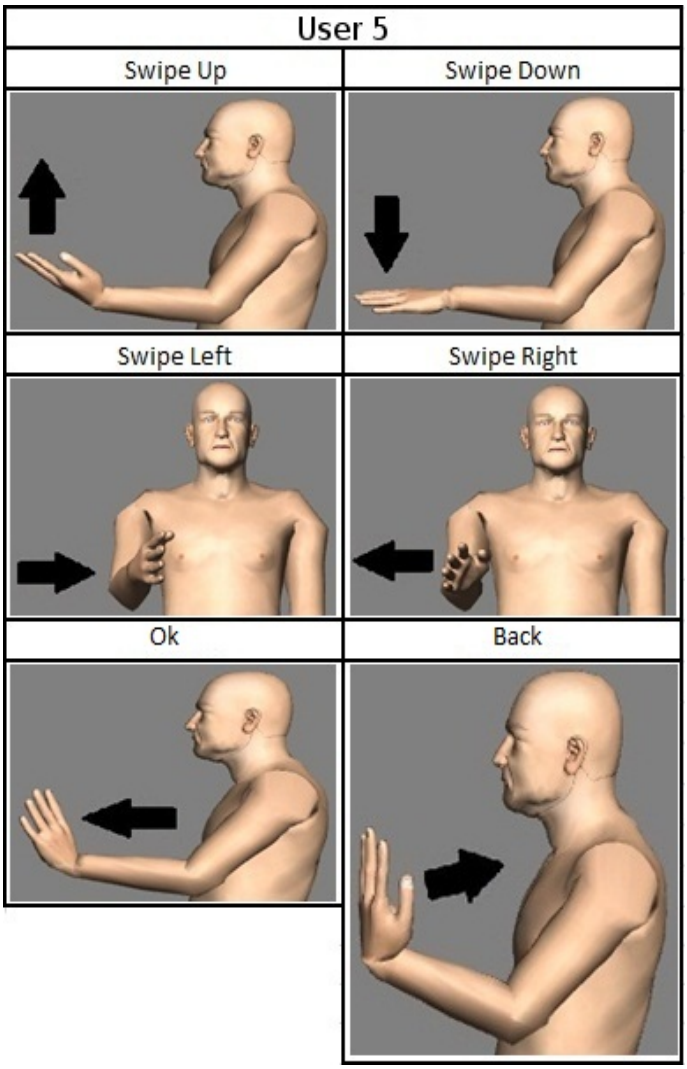

Figure 6.7: Gestures performed by user 5

From the five persons tested, three of them didn't use the hand open like represented on the images. Since Kinteract detects the hand open, closed or in any other position, we don't find it relevant to show that in these last five Figures, but only demonstrate the movement that hand makes to realize the intended event.

6.3 Assessing interaction success

All these five old people were requested to interact with the Kinteract system at the end of these tests, after we have explained which are the gestures defined at that time on Kinteract. The overall reaction was very positive, because they've all managed to navigate throughout the menus without major difficulties, being able by themselves (especially the user 4) to find out that some gestures would only be detected at a particular speed.

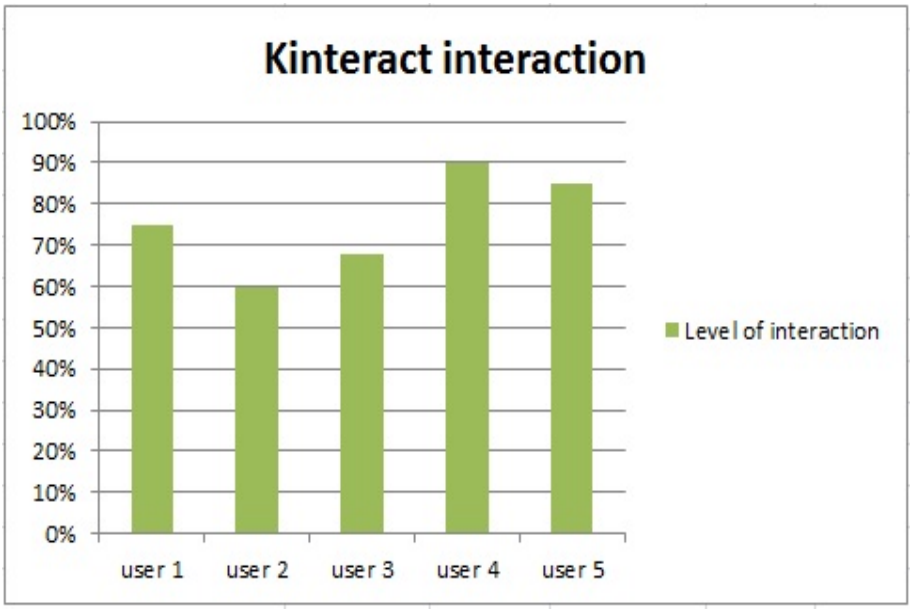

Figure 6.8: Kinteract user level of interaction

This first chart represents the percentage of success that each user had with Kinteract. The values were obtained by visual inspection of the video recording of the tests and this particular chart is related to the third step of the tests. The values were applied in a scale of 0 to 100% and represent the assessment of the success rate of actions that were implemented in Kinteract. This was possible because the gestures defined in Kinteract at this time were already close to the final gestures.

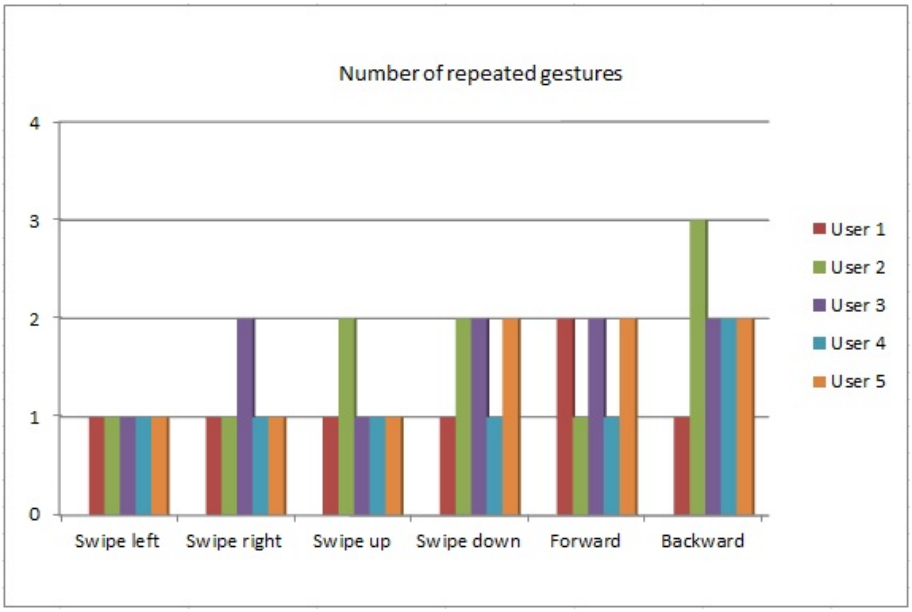

Figure 6.9: Number of repeated gestures to effectively control the eCAALYX user interface

This bar chart represents the number of times that the subjects had to do the gesture to provoke an event on the system eCAALYX user interface. We can conclude then that the most intuitive gestures are "Left", "Right", "Up", "Down" and "Ok", which match the ones implemented in Kinteract. The "Back" gesture wasn't very clear for the old people and Kinteract didn't react properly. The tests demonstrated that the gesture of these people have defined for this movement was a little different than we expected, however with the time that we had tried to approach much as possible.

Chapter 7

Conclusion and Future Work

The system described throughout this document can be a valuable tool that can help the old people to interact with new interfaces.

As the population ages, it is becoming increasingly important to provide older people with a set of conditions that promote a successful ageing. It is also undeniable that the use of technology affords an opportunity to enhance elders quality-of-life through the use of a variety of mechanisms. Whether for information search, health control or just for providing enjoyable experiences, computer systems should be faced by elders as a tool to increase their overall well-being, instead of a world that is completely out of their reach.

Therefore designing systems that meet older adults needs and that consider both their capabilities and limitations, likes and dislikes, have the potential to create artefacts of sheer importance in the contemporary society.

Kinteract wants to act precisely at this point and help the seniors to become more familiar with technology. The main achievement of this project is to create a driver/daemon that processes raw input from the sensor Kinect, identify simple gestures, and injects events in the Linux input layer.

With respect to gestures, and for the purpose of being close to the main target audience, we studied some practical use cases in different areas in which the Kinect sensor was explored. We also had the chance to make a light introduction to the history of natural interaction.

At first, we started to study the frameworks that were available for the specific

operating system that allow operating with the sensor. OpenNI and NITE were the selected platforms, providing a plenty of features that eased the process of development and fastened the practical results.

After assimilating the necessary know-how, we then needed to understand the technical part of Kinect and how it processes raw input from the sensor to the computer. For this project to be successful, we have then defined the gestures that would be necessary to complete the primary use case, i.e., control the set-top box developed for the eCAALYX project.

A detailed specification was the next step and includes a clear definition of our application's requirements and development methodology, in a high level design. A more technical description of Kinteract is also included, going down on a lower level that can reveal some technical aspects of the implementation. Finally, there was the chance to make some field tests and present detailed results. These have revealed that the subjects generally define the "Back" gesture in a way that wasn't the one that we're expecting. The "Back" gesture is working but probably requires a proper support and fine-tuning, unlike all other five gestures that were well defined.

In summary, we were able to identify all the gestures for controlling the set-top box in a general manner, intuitive and simple for the people to whom this project is designed. We consider that the outcome of their work is a valuable contribution and a foundation for future research and work in Gesture recognition for natural interaction with applications.

Building upon this research, a foundation for future research and work arises. It would be interesting to:

- improve the support for "Back" gesture, so this project can be fully adapted to any old person;
- integrate Kinteract into another project in order to test its functionality and make it even more complete;
- use Kinteract to promote physical activity through a body positioning game, asking the player to make the position that is shown on screen. Kinect will be used for the detection of skeletal position in order to validate the position made by the user. This

way, there will be a strong appeal for the mobility of old people, fostering their own health and well-being;

- explore the sensor's microphone array for spoken commands, using the voice as input source.

Finally, there are many ways to keep improving this project, either by expanding the support for the sensor and exploring its powerful features or by adapting the use of Kinteract to different scientific areas.

Appendix A

Acronyms

OS - Operating System

ENIAC- Electronic Numerical Integrator And Computer

PSX - PlayStation X

PS2 - PlayStation 2

PS - PlayStation

PC - Personal computer

CeBIT - Center for Office Automation, Information Technology and Telecommunication

TV - Television

RGB - Red, green, blue

IR- Infrared

VGA - Video Graphics Array

FPS - Frames for second

USB - Universal Serial Bus

LIDAR - Light Detection And Ranging

UK - United Kingdom

SDK - Software Development Kit

DICOM - Digital Imaging and Communications in Medicine

ToF - Time of Flight

LED - Light-Emitting Diode

API - Application Programming Interface

LGPL - Lesser General Public License

GPL - General Public License

GPLv2 - GPL version 2

GUI - Graphical User Interface

FOV - Field of View

NUI - Natural User Interfaces

ASIC - Application Specific Integrated Circuit

LGA - Land Grid Array

CMOS - Complementary metal–oxide–semiconductor

ASCII - American Standard Code for Information Interchange

FPS - Frames Per Second

XML - Extensible Markup Language

References

[1] 'Getting started with OpenNI and X-Box Kinect', 6 June 2011. http://www.uxmagic.com/blog/post/2011/06/06/Getting-started-with-OpenNI-and-X-Box-Kinect.aspx.

[2] L. Spadola, A. Rosset and O. Ratib. Osirix. 'An open-source software for navigating in multidimensional dicom images'. *Journal of Digital Imaging*, 17(3):pp. 205–216, 2004.

[3] Kinect For Windows Blog. 'Microsoft Kinect for windows SDK-v1.0 release notes', 5 February 2012. http://www.microsoft.com/en-us/kinectforwindows/develop/release-notes.aspx.

[4] Kinect For Windows Blog. 'Starting February 1, 2012: Use the Power of Kinect for Windows to Change the Worlds', 9 January 2012. http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx.

[5] MU News Bureau. 'MU Researchers Use New Video Gaming Technology to Detect Illness, Prevent Falls in Older Adults', 6 September 2011. http://munews.missouri.edu/news-releases/2011/0906-mu-researchers-use-new-video-gaming-technology-to-detect-illness-prevent-falls-in-older-adults/.

[6] Calit2. 'Popular Science Highlights Calit2's 'Awesome' Lab for Archaeology', 2 September 2010. http://calit2.net/newsroom/article.php?id=1736.

[7] Jude Divierte. 'CES: PreShow – one quick takeaway from Ballmer's Keynote at CES', 5 January 2011. http://www.whatsyourdigitaliq.com/ces-preshow-my-quick-takeaway-from-ballmer%E2%80%99s-keynote-at-ces/.

[8] A. Kolb, E. Barth and R. Koch. 'ToF-sensors: New dimensions for realismandinteractivity'. *In Computer Vision and Pattern Recognition Workshops*, CVPRW '08. IEEE Computer Society Conference on:pp. 1–6, 2008.

[9] Max Eddy. 'Kinect Hacked for Helpful Robot Shopping Cart', 5 June 2011. http://www.geekosystem.com/wi-go-kinect-hack/.

[10] B. Fernandes and J. Fernández. 'Using Haar-like Feature Classifiers for Hand Tracking in Tabletop Augmented Reality'. *XII Symposium on Virtual and Augmented Reality*, Universitat Politècnica de Catalunya, Departament d'Expressió Gràfica a l'Enginyeria, Natal, RN, Brasil, May 2010.

[11] Jim Giles. 'Inside the race to hack the Kinect', 23 November 2010. http://www.newscientist.com/article/dn19762-inside-the-race-to-hack-the-kinect.html?full=true.

[12] Github. 'OpenKinect / libfreenect'. https://github.com/OpenKinect/libfreenect.

[13] Jianming Guo. 'Hand Gesture Recognition and Interaction with 3D stereo Camera'. *Department of Computer Science, Australian National University*, COMP 8740 Project Report(U4692950):pp. 13–23, November 2011.

[14] A. Industries. 'The Open Kinect project – THE OK PRIZE – get $3,000 bounty for Kinect for Xbox 360 open source drivers', 4 November 2010. http://www.adafruit.com/blog/2010/11/04/the-open-kinect-project-the-ok-prize-get-1000-bounty-for-kinect-for-xbox-360-open-source-drivers/.

[15] PrimeSense Natural Interaction. 'MAKING THE MAGIC HAPPEN'. http://www.primesense.com/nite.

[16] Randolph Jonsson. 'NSK develops four-legged robot "guide dog"', 21 November 2011. http://www.gizmag.com/nsk-four-legged-robot-guide-dog/20559/.

[17] Sean Kean, Jonathan Hall, and Phoenix Perry. *'Meet the Kinect - An Introduction to Programming Natural User Interfaces'.* Apress, New York, new. edition, 2011.

[18] Kinemote. http://www.kinemote.net/.

[19] B. Lee and J. Chun. 'Manipulation of virtual objects in marker-less AR system by fingertip tracking and hand gesture recognition'. *in ICIS '09*, Proceedings of the 2nd International Conference on Interaction Sciences,NY, USA, 2009.

[20] Shane McGlaun. 'Freie Universtat Berlin shows off autonomous electric wheelchair at IFA that uses Kinect', 5 September 2011. http://www.slashgear.com/freie-universtat-berlin-shows-off-autonomous-electric-wheelchair-at-ifa-that-uses-kinect-05177004/.

[21] MESAImaging. 'SR4000 Data Sheet'. Zürich, Switzerland(U4692950):pp. 3, August 2011.

[22] Microsoft. 'Kinect Effect Reaches Into Hospitals, Senior Centers', 19 December 2011. http://www.microsoft.com/presspass/features/2011/dec11/12-19KinectEffect.mspx?rss_fdn=Custom.

[23] Paul Miller. 'Hacked Kinect taught to work as multitouch interface', 11 November 2010. http://www.engadget.com/2010/11/11/hacked-kinect-taught-to-work-as-multitouch-interface/.

[24] OpenNI. 'OpenNI Documentation', 2011,
. http://openni.org/Documentation/home.html.

[25] OpenNI. 'OpenNI Documentation', 2011,
. http://openni.org/docs2/Tutorial/classxn_1_1_context.html.

[26] OpenNI. 'Programmer Guide', 2011,
. http://openni.org/Documentation/ProgrammerGuide.html.

[27] Chris Palmer. 'UCSD Researchers Modify Kinect Gaming Device to Scan in 3D', 28 July 2011. http://www.calit2.net/newsroom/article.php?id=1881.

[28] PMDTechnologies. 'PMD[vision] CamCube 2.0 Datasheet'. Siegen, Germany(U4692950):pp. 3, June 2009.

[29] Frati, V., Prattichizzo, D. 'Using Kinect for hand tracking and rendering in wearable haptics'. *World Haptics Conference (WHC)*, IEEE, 2011.

[30] OpenKinect Project. 'Main Page OpenKinect'. http://openkinect.org/wiki/Main_Page.

[31] Margaret Rouse. 'Virtual reality', June 2009. http://searchcio-midmarket.techtarget.com/definition/virtual-reality.

[32] Ralph Schroeder. 'Defining Virtual Worlds and Virtual Environments'. *"Virtual Worlds Research: Past, Present & Future*, Vol. 1. No. 1(ISSN: 1941-8477), July 2008.

[33] S.E.Jones. 'Surgeons Increasingly Using Kinect Technology to Help with Surgery', 11 June 2012. http://voices.yahoo.com/surgeons-increasingly-using-kinect-technology-help-11442080.html.

[34] Billy Steele. 'SandyStation interactive sandbox uses Kinect to make topography much more interesting', 30 November 2011. http://www.engadget.com/2011/11/30/sandystation-interactive-sandbox-uses-kinect-to-make-topography/.

[35] M. Graw, V. Tronnier, M. Bonsanto, T. Kipshagen and U. G. Hofmann. 'Touch- and marker-free interaction with medical software'. *In R. Magjarevic et al., editors*, WC '09(Springer Berlin Heidelberg):pages 75–78, 2009.

[36] Greg Toppo. 'Video games help autistic students in classrooms', 6 January 2012. http://www.usatoday.com/news/health/story/2012-05-31/video-games-autism-students/55319452/1.

[37] A. Valli. 'The Design of Natural Interaction'. *Kluwer Academic Publishers, MA, USA*, vol.38(no. 3.Hingham):pp. 295–305, July 2008.

[38] Jim Vallino. 'Introduction to Augmented Reality', August 2002. http://www.se.rit.edu/ jrv/research/ar/introduction.html.