UNIVERSITÀ DI PADOVA

FACOLTÀ DI INGEGNERIA

TESI DI LAUREA

A COMPARISON BETWEEN GESTURE TRACKING MODELS, AND THE DEVELOPMENT OF AN INTERACTIVE MOBILITY AID SYSTEM FOR THE VISUALLY IMPAIRED

Laureando: Nicola Scattolin

Relatori: Prof. Sergio Canazza, Prof. Marc Leman
Correlatori: Prof. Giovanni De Poli, Prof. Antonio Rodà
Controrelatore: Prof. Nicola Orio

Corso di Laurea Magistrale in Ingegneria Informatica

Data Laurea 16-07-2012

Anno Accademico 2011/2012

Alla mia famiglia

Contents

Abstract

This thesis covers the study, the definition, and the setup of systems to track the human gesture, and the realization of a device to assist visually impaired people during the exploration of known or unknown environments.

It has been started at Institute for Psychoacoustics and Electronic Music (IPEM) at the University of Gent, where a particular research area is dedicated to the study of the role of the human body in relation with all musical activities. IPEM researchers have several type of sensors available to use for this purpose: one of them is a Motion Capture system (*MoCap*), installed at IPEM laboratory. *MoCap* is a very expensive and cumbersome system to capture peoples' motions, on the other hand it is very precise and accurate in gathering data. The release of Microsoft *Kinect* sensor has caught researchers attention as a possible substitute to *MoCap* being a cheaper and more portable device for all the experiments where costs and installation spaces are problematic. *Kinect* sensor is clearly not accurate as *MoCap* system, in fact this thesis investigates the differences and the errors that the device from Microsoft does to estimate positions and movements of a person, taking as reference system the trustworthy IPEM's *MoCap*.

At the end of the evaluation stage of the device, the thesis is continued at Centro di Sonologia Computazionale (CSC) at the University of Padova, where an application has been developed in order to help visually impaired people. The system - called *SoundingARM* - is usually installed in the middle of a room, in combination with *Kinect* sensor which is turned towards the entrance door, ready to recognize every incoming people. As soon as a person goes in, he can recognize obstacles or a piece of furniture just moving his arms: *SoundingARM* is able to identify the movement that a person naturally does to point an object and it is able to report the name of pointed object by text-to-speech synthesizer. So by this device visually impaired users are able to reconstruct themselves a mental map of the room, without having to go throw the door.

In conclusion this work contains wide documentation about obtained results during the experiments for the evaluation of *Kinect* device and a description of development and implementation stages of *SoundingARM* application.

Chapter 1

Introduction

Main topics of this thesis are the analyses of motion capture systems, definition, and development of a device to aim visually impaired people during exploration of a familiar or an unknown environment. Thesis work has been started at University of Gent, at Institute for Psychoacoustics and Electronic Music (IPEM), where a very accurate and rather expensive device - called *MoCap* - is installed for tracking people's movements and behaviors.

The presence at IPEM of a so specific and expensive technology may not surprise, indeed it comes in useful to study correlation between human behaviors and movements and music: Prof. Marc Leman, director of IPEM, and Prof. Rolf Inge Godøy ask themselves if musical gestures and behaviors are related to the music [1].

> “Why is it to many listeners are able to spontaneously make gestures that seems to fit the music? Why do they make these gestures? Furthermore, how are these gestures related to the music, and how are these gestures related to gestures of performers? Or in general, what are gestures? And how do gestures function in contexts of music performance and listening? If we assume that music communicates movement, where can we find movement in sound, or what does it mean that sounds contain movement? If gestures express an idea or meaning, what kind of idea or meaning is it? ”

They theorize that musical experience is inseparable from the sensations of movement, and hence, that studying these gestures, what they call *musical gestures*, ought to be a high priority task in music research. The integration of gestures with perception and with thinking in general is labeled “embodied cognition”, including insights on how body movement is both a response to

whatever is perceived and an active contribution to our perception of the world. The study of musical gestures appears as a core area of modern music research, with links to engineering, neuroscience and both human and social science.

A challenge is to develop sensor and computer vision solutions, and corresponding computational algorithms, which understand the gestures (here used in the communication sense) in a continuous stream of movement. While humans have few problems separating a hand gesture (e.g. waving goodbye) from other types of movement (e.g. waving away a fly), this is much more problematic for computers. This is only due to the remarkable of visual scene analysis in humans, but it also due to the fact that we understand the intended meaning of the gestures gesture based on its context and on our life-long experience of multimodal communication.

The study of motion with respect to music started from an intuitive point of view involving music analysis phenomenology and hermeneutic interpretation. After 1920, experimental methods and empirical data based on observation and measurement were taken into account, psychologist began collecting great amount of data on attitudes in children; in social groups; towards racial minorities; and of many different populations in varying situation. Sophisticated sampling procedure were devised, and scales were constructed so attitude intensity could be assessed [2]. Recently, scientific and computational methods have been included into research work on music, motion and gestures [3, 4, 5, 6].

The cost of a motion capture system (*MoCap*) is of several thousands of Euros, but since November 2010 a very cheaper device could be bought in a normal supermarket. Its name was Microsoft *Kinect*, it was created as a game device to connect to Microsoft Xbox 360 ®, and it allowed people control games with their body movements. As much rapidly worldwide researchers foresaw new possibilities for their experiments, and a even larger group of people began to develop application for computer, using earlier driver, developed by opensource community.

Nevertheless the *Kinect* was created for games, technically is a 3D scanner, in fact, it transforms what it sees in a depth matrix, that matrix could be seen as an "depth-image" mapping the depth values in a range of colors. *Kinect* drivers are able to detect and track the skeleton up to 4 users, similarly to a expensive *MoCap* system, but it costs ten times less. One of the aims of this work is study the accuracy of *Kinect* and the feasibility to substitute the *MoCap* system in some experiments. To evaluate the *Kinect* device it was taken as reference system the *MoCap* system, in fact, its trustworthiness was already tested in IPEM experiments.

1.1 Embodied Music Cognition

Embodied music cognition is a direction within systematic musicology interested in studying the role of the human body in relation to all musical activities. It considers the human body as the natural mediator between mind (focused on musical intentions, meanings, significations) and physical environment (containing musical sound and other types of energy that affords human action) [7].

Embodied music cognition tends to see music perception as based on action. For example, many people move when they listen to music. Through movement, it is assumed that people give meaning to music. This type of meaning-formation is corporeal, rather than cerebral because it is understood through the body. This is different from a disembodied approach to music cognition, which sees musical meaning as being based on a perception-based analysis of musical structure. The embodied grounding of music perception is based on a multi-modal encoding of auditory information and on principles that ensure the coupling of perception and action.

During the last decade, research in embodied music cognition has been strongly motivated by a demand for new tools in view of the interactive possibilities offered by digital media technology. With the advent of powerful computing tools, and in particular real-time interactive music systems like *Max/MSP* or *Pure Data*, gradually more attention has been devoted to the role of gesture in music. This musical gestures research has been rather influential in that it puts more emphasis on sensorimotor feedback and integration, as well as on the coupling of perception and action. With new sensor technology, gesture-based research has meanwhile become a vast domain of music research, with consequences for the methodological and epistemological foundations of music cognition research. Research in embodied music cognition has a strong connection with technology development, more particularly, in fields related to interactive music systems, and music information retrieval. Mediation technology is the technology by which the human body, and consequently also the human mind, can be given an extension in the digital musical domain [8].

Motion capture devices allow to capture movements or general actions of a single person or a group of persons. These devices product data streams those could be analyzed in real time, for example using *Max/MSP* or *Pure Data*, or offline, using for example *Matlab*.

The real time analysis allows the achievement of real time application like games, how it happens in *Sync-In-Team*, a music game developed at IPEM [9], where two groups of participants obtain different score and video feedback in relation to how they are synchronized with the music.

Another real time example is the *Stanza Logo-Motoria*, developed at Uni-

versity of Padova [10]; it is a multimodal interactive system for learning and communication developed by means of the EyesWeb XMI platform. It is permanently installed in a Primary School where it is used as an alternative and/or additional tool to traditional ways of teaching. The *Stanza Logo-Motoria* is used by all the pupils of the school - from the first to the fifth grade - including the children with disabilities. It is a technological tool to enhance alternative intelligences and communication, and its purpose is to promote learning motivation and develop pupils' different cognitive styles.

1.2 Application *SoundingARM*

The construction of environmental cognitive maps of spaces is fundamental for orientation and mobility skills development. Since the visual channel gathers most of the spatial information, people with severe visual impairments, who are partially or totally unable to see, face difficulties in: moving in medium-scale spaces, where the locomotion is needed for exploration, immediately recognizing the type of an indoor environment or rapidly finding an object.

Assuming that the support of appropriate spatial information by means of compensatory sensory channels may contribute to blind people's spatial performance, now it is introduced a non-invasive system, *SoundingARM* (Acoustic Representation of a Map), which is able to quickly offer an essential acoustic map of an (known and unknown) indoor environment. By means of this system the user can promptly explore a room by standing in the doorway and by performing a simple gesture (the finger pointing).

1.2.1 Related works

In the latest years, the use of more and more advanced auditory display techniques have increased the possibilities to compensate the lack of vision which affects millions of blind and low-sighted people in the world. Auditory displays can aid blind people in orientation and mobility tasks. The aid devices for the blind people usually use speech synthesis techniques [11]. There are a lot of devices that offer information by talking to the user, first of all, the screen-reading software; the commonly used are: JAWS, Window-Eyes, VoiceOver, SAToGO, NDVA, and ZoomText.

Other talking devices include reading machines, from portable to desktop solutions and computer-based to standalone solutions, which consist of a document scanner, a OCR software, and a speech synthesizer: for instance, ALLREADER and EYE-PAL. Recently, it can easily find also applications, such as Voice Brief, a text to speech voice assistant for email and other texts for iPhone, iPod touch,

and iPad.

To extend the roundup on the talking devices, it is focused on the "way-finding technologies" which can be subdivided into two main categories: outdoor and indoor systems. Generally, outdoor systems are based upon Global Positioning System (GPS) to locate the user: for example, the Atlas system [12], a digital talking map GPS-based, which provides a verbal information on locations, travel directions, and points of interest. Outdoor systems can rely also upon the infrared communication, for instance, Talking Signs®[13] that consists of an emitter permanently installed in the environment and a hand-held receiver.

The indoor systems, indeed, typically depends on infrared (IR) [14], ultrasound [15], radio frequency identifier (RFID) tags [16], or computer vision [17]. Nowadays, also the mobile technology is delivered with applications for navigation: by combining GPS data with the data from a magnetometer, directional information can be offered to a user when the device is pointed in a specific direction [18]. German Navigational Aid for Visually Impaired (NAVI) system, assembled at University of Konstanz, utilizes a *Kinect* as input device, instead of simple cameras [19]. For detecting the immediate surroundings, German researchers reversed the standard operating principle of the *Kinect*. Instead of a static *Kinect* that tracks moving objects, they track the static environment with a moving head-mounted *Kinect*. A 12V battery pack was utilized to power the mobile *Kinect* that lasted for about 5 hours. A vibrotactile output is provided by a waist belt that contains three pairs vibe boards. The speech output is provided by an ordinary Bluetooth headset for mobile phones. Canadian researchers built a similar system for mobility of impaired people, in a helmet is mounted the *Kinect* sensor, and inside, a vibrotactile belt. The obstacles are signaled to the user with vibrotactile signals in his forehead, the vibration increases in inverse proportion to the distance until a continuous pressure against user's forehead to signal him an impending collision with an obstacle [20]. In all these cases the user has to navigate into the environment, and wear/hold a sensor in order to receive spatial information.

1.2.2 *SoundingARM*

Unlike the systems briefly illustrated above, *SoundingARM* allows the user to quickly explore an indoor space by standing in the doorway and by performing a simple gesture (the finger pointing) without wearing sensors. These two simple tasks aid blind people in development of an auditory map which can be used to immediately recognize the type of indoor environment, to safer move in space, or to quickly detect a specific object.

SoundingARM, was developed using the Microsoft *Kinect* device, thought to be non invasive, cheaper, easily to use and to install in a indoor environment, its aim is helping partially sighted and visually impaired people during the exploration of a domestic environment, for example their living room, in order to make them easier going home after a stay in hospital, or to explore quickly an unknown environment, for example a hotel room.

With impairment or failure of vision, blind people must shift their attention to other senses to obtain information about the environment. Those who are totally blind depend entirely on hearing, touch, smell, and the vestibular sense to perceive, interact with, and move about their environs.

After vision, hearing has the broadest band for acquiring information. Blind people rely almost entirely on hearing to perceive the environment beyond their reach. The idea that blindness leads to a perceptual compensation that is manifested as an over-performance of hearing has found support in cortical electrophysiology [21, 22], functional brain imaging [23, 24], and behavioral studies [25, 26]. Auditory compensation might consist of a reorganization and reallocation at the level of the cortex (structural hypothesis) so that auditory and tactile areas function better or are the result of a better development due to the vision impairment (strategic hypothesis) [27]. Most of the visually impaired were educated to navigate according to their listening skills in their everyday life.

Furthermore, blind people have more experience attending to, and interpreting environmental auditory information than do blindfolded sighted people, the usual comparison group [11]. Hearing is a sense which allows different levels of intenseness of perception, where the range reaches from background sound to speech. For example in [28] is presented a system which allows computer users to explore a virtual environment only by their sense of hearing, visually impaired people should be able to explore computers as they do explore rooms they do not know in their everyday life, enabling visually impaired people to get a first overview of an environment without the necessity to explore it in detail.

SoundingARM is able to quickly offer an essential acoustic map of an (known and unknown) indoor environment. By means of this system the user can promptly explore a room by standing in the doorway and by performing a simple gesture (the finger pointing).

A purpose of exploring a room might be, for instance, to look for an object. Usually, two specific types of independent search patterns are used when exploring an area: the exploration following the perimeter which provides information about the size and shape of the area and or the exploration by means of a series of straight-line movements to and from opposite sides of the perimeter. It is simple to imagine that all these strategies take a lot of time of locomotor

exploration. The *SoundingARM* system aims at offering users with an auditory "first sight" of space, an acoustic map of space, allowing them to immediately be aware of the environment type and of the objects in it.

If the user points his/her arm around to find an object, the system answers giving the audio feedback of it. Moreover, the user obtains the position of the object information by the gesture itself: the object is located in the direction on which the finger is pointed. At the time, the audio feedback is constituted by a Text-to-Speech synthesis of the object description, or simply the name of the object. Future development will introduce 3D sound spatialization in order to approach an auditory visual system.

Chapter 2

Motion Capture Systems

Motion capture (*MoCap*) is sampling and recording motion of humans, animals, and inanimate objects as 3D data. The data can be used to study motion or to give an illusion of life to 3D computer models. Since most *MoCap* applications today require special equipment there are still a limited number of companies, schools, and organizations that are utilizing *MoCap* technology. However, most people, even small children, have seen the films, games, and TV commercials for which *MoCap* technology is used.
MoCap systems commercially available today can be categorized into three main groups: optical systems, magnetic systems, and mechanical systems [29].

2.1 Optical *MoCap* systems

A typical optical system consists of 4 to 32 cameras and a computer that controls the cameras. With most optical systems capture subjects wear markers, where markers are either reflective (passive) or light emitting (active). Passive markers are made of reflective materials and their shapes are spherical, semi-spherical, or circular. Shapes and sizes of markers depend on the camera resolutions and capture subjects (e.g., smaller markers are used for facial and hand captures). Passive markers are attached directly to a capture subject's skin or Velcroed to a *MoCap* suit, which is a full-body unitard made of stretchy materials, such as spandex. Cameras (Fig. 2.1a) in a passive marker system are equipped with light-emitting diodes (LEDs) and the lights emitted by the LEDs are reflected by markers On the other hand, markers in an active marker system are LEDs. Some active marker systems illuminate one LED at a time, eliminating the need for identifying each marker. Others illuminate all LEDs at once. Modulating the amplitude or frequency of each LED allows such systems to identify markers. Some of the latest active marker systems work in natu-

ral lighting conditions, that is, they can capture subjects in various costumes at locations outside studios; however, lighting must be carefully controlled for most optical systems, especially passive marker systems. Cameras in an optical system capture the lights reflected or emitted by markers at speeds somewhere between 30 and 2000 Hz. At least two cameras need to see a marker in order to determine the 3D position of the marker, although three or more are preferred for accuracy. Sometimes a capture subject herself/himself, another capture subject, or a prop hides (occludes) some of the markers on the subject. For instance, when a subject lies flat on the stomach, the markers on the subject's front will be occluded. When markers are occluded, no camera sees them and it results in loss of data. There are data editing techniques and tools to make up for missing data but when too many markers are occluded or the duration of an occlusion is too long, it is impossible to fix the problem. Optical data generated by a state of the art system is very clean and accurate when it does not suffer from occlusion problems.

Marker configurations are flexible with optical systems. You can use the marker configurations that the system manufacturer provides you with or you can design your own that suits your needs. A relatively large number of markers can be tracked simultaneously, for example, up to 200 markers with a 16 camera system. Since capturing multiple subjects at once tends to cause occlusion problems, capture one subject at a time if it is not crucial to capture multiple performers together. When performers interact with each other and the synchronization among them is important, capture multiple subjects simultaneously. Capture subjects can move freely in a capture volume because no equipment or wires are connected to them.

Optical systems' real-time visual feedback during capture is often limited to stick figures, although linking a *MoCap*'s real-time output to a specific real-time application such as *MotionBuilder*[1] will render real-time results. Other systems such as the *Giant Studios* system readily render real-time characters directly in system. Recorded data is still processed to compute the trajectories of the markers in a rather extensive post-processing to get the best, most stable results. Rotational data can be computed in real time, but is usually computed from positional data in post-processing.

Among the markerless *MoCap* technologies that recently emerged, Mova's Contour Reality Capture system is an optical system that captures the continuous skin surface of a moving capture subject, instead of a small number of points on a capture subject. A capture subject wears a phosphorescent makeup and two sets of cameras capture the texture and geometry of the subject in a movement [29].

[1] MotionBuilder® is a 3D character animation software by Autodesk®

2.2 Magnetic *MoCap* systems

Magnetic (electromagnetic) *MoCap* systems are sometimes called magnetic trackers. The systems were derived from the sensors placed on a military aircraft pilot's helmet to track the pilot's head position and orientation for the helmet-mounted display. With a magnetic *MoCap* system, 12 to 20 tracking sensors are placed on a capture subject to measure spatial relationship to a magnetic transmitter. The tracking sensors output their translations and orientations. Hence, no postprocessing is required to compute rotations. This fact allows magnetic systems to be used for realtime applications.

Tracking sensors are not occluded by capture subjects or props made of non-metallic materials, which is an advantage over optical systems. However, they are prone to magnetic and electrical interferences caused by metal objects and electronics in the environments. Interferences can result in distorted output. Building structures with highconductivity metals are not suitable as capture spaces for magnetic systems. The wiring and batteries for tracking sensors may limit capture subjects' movements. Moreover, tracking sensors' batteries need to be recharged every few hours. Magnetic systems can be divided into two groups. One group uses direct current electromagnetic fields and the other uses alternating current fields. AC systems are very sensitive to aluminum and copper, while DC systems are sensitive to iron and steel.

Magnetic systems' sampling rates (up to 144 or 240 Hz) are lower than optical systems and magnetic data tends to be noisy. Tracking sensors' configurations cannot be changed as freely as optical systems' marker configurations. Magnetic systems can capture multiple performers simultaneously with multiple setups. Magnetic systems' capture volumes are normally smaller than optical systems. One of the biggest advantages of magnetic systems is their cost, magnetic systems are less expensive than optical systems [29].

2.3 Mechanical *MoCap* systems

Mechanical (exo-skeletal) *MoCap* systems directly measure joint angles of a capture subject who wears an articulated device that consists of straight rods and potentiometers. Straight rods are linked with potentiometers at the joints of the body, designed to measure joint angles as the capture subject moves. The device looks like an exo-skeleton. Other types of mechanical systems include data gloves and digital armatures.

Mechanical systems are real time, relatively inexpensive, free of occlusion, free from magnetic or electrical interferences, and highly portable. Wireless mechanical systems provide large capture volumes. A notable disadvantage of

mechanical systems is that they do not measure global translation very well. They measure it using accelerometers, but the data can still "slide" and "slip" a little. They do a poor job when the feet leave the floor. If a capture subject jumps up, the data will normally not follow the jump and the data will stay on the floor. If a character walks up stairs, the data will never go up in the air but look as if it were walking in place. Magnetic sensors are often added to mechanical systems to correct this problem. The joints in articulated exo-skeletal systems are simple hinge joints, although we, humans, have other kinds of joints, such as ball and socket joints, gliding joints, saddle joints, and pivot joints. This means that articulated exo-skeletal systems restrict capture subjects' movement at their joints. Also the device's volume and breakability restrict subjects' movement, for example, a capture subject wearing an articulated exo-skeletal system probably doesn't want to roll around on a floor since it hurts and breaks the device [29].

2.4 IPEM's *MoCap* system

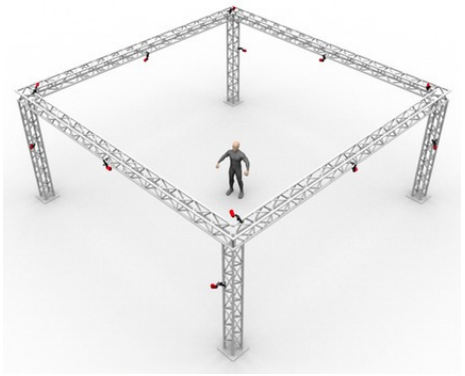
The *MoCap* system installed in IPEM laboratory is a "Optical" one, produced by *OptiTrack*. It is constituted by 12 infrared cameras (fig. 2.1a), each camera has got a constellation of infrared LEDs around its objective. These cameras are connected with a computer through USB cables, the computer has a software called *Arena* that manage the received data from all the cameras and calculated the 3D positions of all the markers. This software helps to define rigid bodies, i.e. a group of markers can be grouped and tagged, in order to have more reference in the data collections. *Arena* can handle more than one human skeleton and stream their data to other software allowing real-time data elaboration from other computers. Lastly the application allows the export of data in standardized 3D file, as C3D and BVH.

Before starting the experiments, the system has to be calibrated following a procedure that takes more or less 20 minutes. After it the system can be reused anytime except if the cameras were been shifted. However it is better to do any event, even if the cameras were not moved, because during the procedure it is possible to prevent the occlusion of a camera or find some false markers (every reflective surface could be seen as a marker by the system).

In appendix C is reported an example of a setup procedure for IPEM laboratory's *MoCap* executed every time before an experiment.

(a) An *OptiTrack* camera, used by the *MoCap* system installed at IPEM. IR LEDs surround a infrared sensor (like a camera, in the middle)

(© 2011 NaturalPoint, Inc.)



(b) Rapresentation of the position of 12 cameras in a *MoCap* system, like IPEM one.

(© 2011 NaturalPoint, Inc.)

(c) A reflective marker.

(© 2011 NaturalPoint, Inc.)

(d) Motion Capture suit, markers can be attached to any of the velcro-friendly surfaces.

(© 2011 NaturalPoint, Inc.)

Figure 2.1: *MoCap* system installed at IPEM

2.5 Microsoft *Kinect* device

Kinect is based on software technology developed internally by Rare, a subsidiary of Microsoft Game Studios owned by Microsoft, and on range camera technology by Israeli developer PrimeSense, which interprets 3D scene information from a continuously-projected infrared structured light. This 3D scanner system called Light Coding employs a variant of image-based 3D reconstruction [30].

It was introduced only as a game device connectible to the Microsoft Xbox 360, in order to follow the trend introduced by Nintendo Wii and PlayStation MOVE / EYE and to simplify the controller of the video game consoles increasing the motion human interaction. In fact, with this new devices, a long sequence of button presses could be substituted by defined movements of a wand.

Microsoft with the *Kinect* seems to move a step over the competitors, the device and its software are able to understand what the player is doing without the support of a wand or other sensors.

The *Kinect* sensor lets the computer directly sense the third dimension (depth) of the players and the environment, making the task much easier. It also understands when users talk, knows who they are when they walk up to it, and can interpret their movements and translate them into a format that developers can use to build new experiences. *Kinect*'s impact has extended far beyond the gaming industry. With its wide availability and low cost, many researchers and practitioners in computer science, electronic engineering, and robotics are leveraging the sensing technology to develop creative new ways to interact with machines and to perform other tasks, from helping children with autism to assisting doctors in operating rooms. Microsoft calls this the *Kinect Effect*.

The *Kinect* sensor incorporates several advanced sensing hardware. Most notably, it contains a depth sensor, a color camera, and a four-microphone array that provide full-body 3D motion capture, facial recognition, and voice recognition capabilities [31].

Fig. 2.2 depicts a schematic top view of device. An illumination assembly (30) comprises a light source (34) (which may be a point source, such as a laser, without additional optics, as explained below) and a transparency (36), which are used in combination to project a pattern of spots onto object (28). The positive image on transparency is an image of the pattern that is to be projected onto object. A single, stationary transparency, fixed in the housing of assembly, with a fixed, uncorrelated pattern of spots, is sufficient for the purposes of 3D mapping (Fig. 2.3a).

Light source transilluminates transparency with optical radiation (infra red)

so as to project an image of the spot pattern that is contained by the transparency onto a object. The light source is a point source, meaning that the rays of radiation emitted by the light source emanate from a locus small enough so that the spot pattern on transparency is replicated sharply on a object. For this purpose, light source comprises, for example, a coherent source with large angular divergence, such as a laser diode. When a point source is used with the transparency in this manner, no other projection optics are required.

An image capture assembly (32) (see figure 2.2) gathers an image of the pattern that is projected by illumination assembly onto a object. Assembly comprises objective optics (40), which focus the image onto an image sensor (42). Typically, sensor comprises a rectilinear array of detector elements (44), such as a CCD or CMOS-based image sensor array. Assembly comprises a bandpass filter (not shown in the figures), chosen and positioned so that sensor receives only light in the emission band of light source, while filtering out ambient light that might otherwise reduce the contrast of the image of the projected pattern that is captured by the sensor.

The *Kinect* sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone running proprietary software, which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

In the *Kinect*, illumination assembly 2.3b and image capture assembly (fig. 2.2) are held in a fixed spatial relation. This configuration and the processing techniques used by image processor (not in the figure, but it is a simple processor embedded in device) make it possible to perform 3D mapping using the single image capture assembly, without relative movement between the illumination and image capture assemblies and without moving parts.

To simplify the computation of the 3D map and of changes in the map due to motion of a object, illumination assembly and image capture assembly are mounted so that an axis passing through the centers of the entrance pupil of image capture assembly and the spot formed by light source on transparency is parallel to one of the axes of image capture sensor (taken for convenience to be the X-axis, while the Z-axis corresponds to distance from device).

Specifically, a Z-direction shift of a point on the object, δZ, will engender a concomitant transverse shift δX in the spot pattern observed in the image. Z-coordinates of points on the object, as well as shifts in the Z-coordinates over time, may thus be determined by measuring shifts in the X-coordinates of the spots in the image captured by *Kinect* relative to a reference image taken at a known distance Z. Y-direction shifts may be disregarded. This sort of triangulation approach is appropriate particularly in 3D mapping using uncorrelated

patterns of spots, although aspects of the approach may be adapted for use with other types of patterns, as well. Thus, to generate the 3D map of a object, image processor compares the group of spots in each area of the captured image to the reference image in order to find the most closely-matching group of spots in the reference image. The relative shift between the matching groups of spots in the image gives the Z-direction shift of the area of the captured image relative to the reference image [32]. The shift in the spot pattern may be measured using image correlation or other image matching computation methods that are known in the art [33].

The *Kinect* sensor's microphone array enables the Xbox 360 to conduct acoustic source localization and ambient noise suppression, allowing for things such as headset-free party chat over Xbox Live. The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and the *Kinect* software is capable of automatically calibrating the sensor based on gameplay and the player's physical environment, accommodating for the presence of furniture or other obstacles. Through reverse engineering efforts, it has been determined that the *Kinect* sensor outputs video at a frame rate of 30Hz. The RGB video stream uses 8-bit VGA resolution (640×480 pixels) with a Bayer color filter, while the monochrome depth sensing video stream is in VGA resolution (640×480 pixels) with 11-bit depth, which provides 2.048 levels of sensitivity. The *Kinect* sensor has a practical ranging limit of $1.2-3.5$m distance when used with the Xbox software. The area required to play *Kinect* is roughly 6m^2 , although the sensor can maintain tracking through an extended range of approximately $0.7-6$m. The sensor has an angular field of view of 57° horizontally and 43° vertically, while the motorized pivot is capable of tilting the sensor up to 27° either up or down. The horizontal field of the *Kinect* sensor at the minimum viewing distance of 0.8m is therefore 87cm, and the vertical field is 63cm, resulting in a resolution of just over 1.3mm for pixel. The microphone array features four microphone capsules and operates with each channel processing audio at a sampling rate of 16kHz, and at 16-bit resolution [30].

2.5.1 The world seen by *Kinect*'s "eyes"

Microsoft *Kinect* is constituted by several hardware components, but the most important ones are the input devices: a normal camera, a depth sensor and 4 microphones.

The outputs of the cameras are constituted by two matrices, after processing by device, one for the RGB camera and another for the depth camera. The first

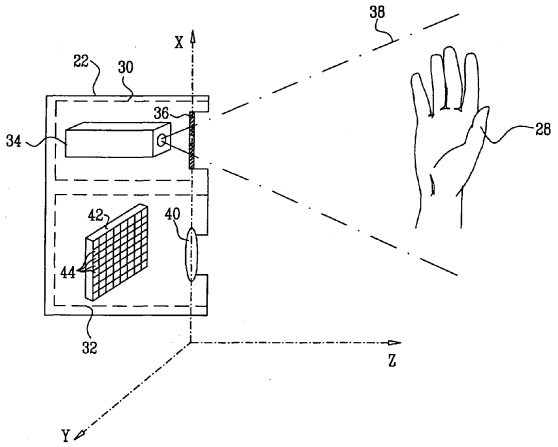


Figure 2.2: Schematic top view of device. 22. *Kinect* device, 28. Object (Hand), 30. Illumination assembly, 32. Image capture assembly, 34. Light source, 36. Transparency, 38. Range of work, 40. Objective optics, 42. Image sensor, 44. Rectilinear array of detector elements.

matrix could be shown as it is in a graphical application. The second matrix is usually utilized for skeleton recognition, it can be mapped in false colors as an image, assigning different shades to the depth values, this feauture could be seen if Fig. 2.4, obtained making a screen shot of *OpenNI NiViewer* application.

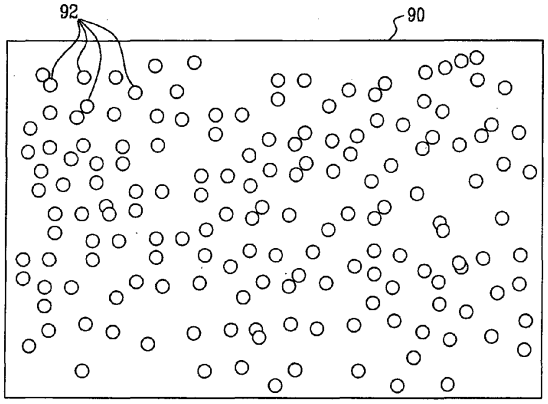
2.5.2 Available software

There are several possibilities to connect the *Kinect* to a computer: Microsoft *Kinect* SDK, running only with Windows 7 and closed source, OpenNI and OpenKinect[34] frameworks, multi-platform and open source. When this work started Microsoft *Kinect* SDK was not been released, in addition, IPEM's researchers preferred to use Mac OS X operative system, hence they chose OpenNI.

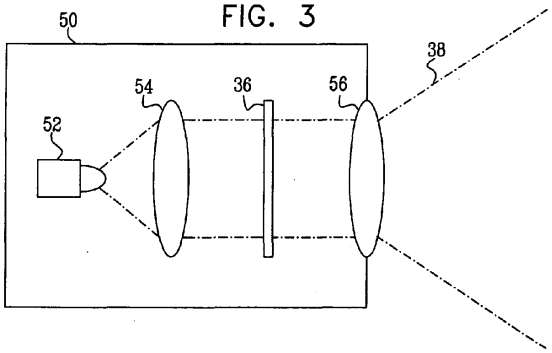
OpenNI project is divided in 3 parts: driver for the sensor (*Kinect*), OpenNI framework and *NITE*. The Drivers allows the operative system to recognize the device, the OpenNI framework is an intermediary because it helps application developers offering a set of API and at least *NITE* middleware interacts with OpenNI in order to manage the user's skeleton recognition.

2.5.3 Accuracy estimation experiments

To estimate accuracy of *Kinect* were settled 3 main experiments, they involve use of both *OptiTrack* system of IPEM lab and *Kinect*, using *OptiTrack* system as reference system. The base idea is: if *Kinect* and *OptiTrack* are capturing the same things, and if *Kinect* is almost good as *OptiTrack* the found difference



(a) Schematic illustration of a transparency. 90. Transparency, 92. The spots on transparency comprises micro-lenses.



(b) Schematic top view of an illumination assembly. 36. Transparency, 38. Range of work, 50. Assembly, 52. Light-emitting diode (LED) 54, 56. Suitable optics.

Figure 2.3: Details of illumination assembly.

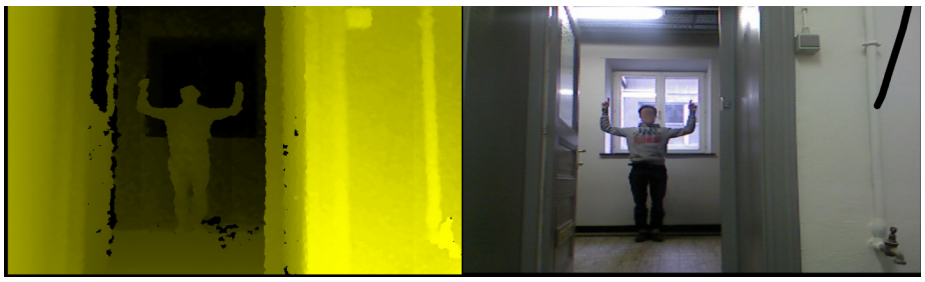


Figure 2.4: The two video inputs of the *Kinect* device, on the left the depth sensor output on the right the web camera one.

in data should be close to zero.

It was taken for granted that *OptiTrack* is better than *Kinect*, that is an obvious choice because the two systems have some abyssal difference: while *OptiTrack* is constituted by 12 IR-camera placed in order to have a view of 360°, *Kinect* has only a depth sensor. Another difference is in the amount of data collected: in *OptiTrack* is 3 times more than the information collected by *Kinect*, in fact, *Kinect* outputs data with a velocity of 30 Hz, instead the 100 Hz of *OptiTrack*.

The setup of all the experiments presume that *Kinect* and player were always in the range of work of *OptiTrack* system. The player was recognizable by the system because he was wearing a *MoCap* suit and *Kinect* was made recognizable thanks to 3 markers put on its top.

The two system are not directly compatible, that means before starting the analysis of the data, some additional calculation have to performed: for example re-sampling, synchronization of the data, reduction to a common reference system (the *OptiTrack* one) and removal of systematic error due to difference between skeleton versions utilized by *MoCap* and *Kinect*. Below the experiments are presented in short, whose are been explained in further chapters:

- In the first experiment it was tried to understand the accuracy of *Kinect* when a user is executing some free movements, for example hands movements, twirl, steps.
- In the second experiment the player was performing some pseudo-random movements, like first experiment, but it was taken data from different distance from the *Kinect* Sensor.
- In the third experiment was evaluated the delay of the *Kinect*.

During the experiment about the delay of the *Kinect* several problems on the capturing of clap movements were found, hence further experiments were done to give an explanation about this weird phenomenon.

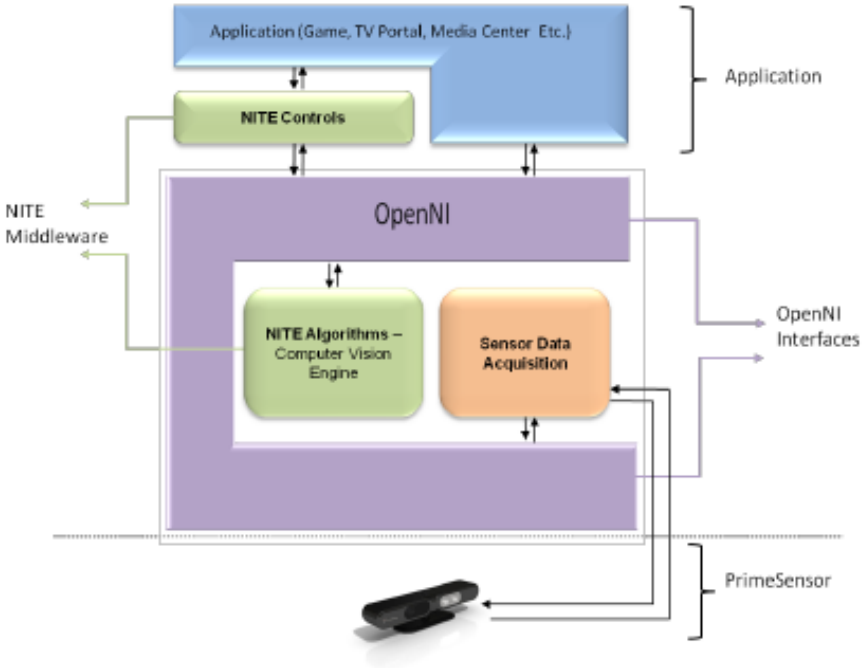


Figure 2.5: The illustration depicts the relationship between the OpenNI framework + *NITE* and the application level. This illustration was taken from [35].

Chapter 3

Data analysis and processing

The *Kinect* is completely different from *OptiTrack* in the hardware and in the software, so also the data structures used by the systems to send or store information. But, at the same time, the use of similar concepts is not misleading because, in the matter of fact, the two systems make the same thing: tracking a human person. For example, *Arena*software, control management application for the *MoCap* system, is able to export the data in BVH (Biovision Hierarchy) file format, a known standard developed by Biovision, to export skeleton information as well as motion data. This file could be easily imported in *Matlab* to be analyzed. With the *Kinect*, *OSCeleton* was used to capture the data stream from the sensor, and a *Max/MSP* patch to write the received information into a simple textfile. Lastly a custom parser was developed to translate the data into a *Matlab* usable file.

In addition, before starting the analysis, the data had to be transformed in order to make it comparable: these operations can add errors or noise to the original data. In the next pages a discussion about the excellence of the data manipulations and transformations will be presented.

Materials

Details of necessary instrumentation to make the experiments:

- *Kinect*
- iMac Desktop to which the *Kinect* is connected.
  - Software:
    - SensorKinect-Bin-MacOSX v. 5.0.1.32 (driver *Kinect*)
    - OpenNI-Bin-MacOSX v. 1.1.0.41 (18 April 2011) (Middleware)
    - PrimeSense NITE Unstable Build for MacOSX 10.6 Universal x86/x64 (32/64-bit) v1.3.1.5 (vs 18, April 2011) (Skeleton Recognition)
    - *Sensebloom / OSCeleton* v. 01 February
    - Max/MSP patch developed by IPEM, with which you can record to a file all the data sent by *OSCeleton* and record an audio stream captured by the microphone installed on computer.
    - Max/MSP v 5.1
  - Hardware e SO: Intel Core 2 Duo 2.4 GHz, 4GB 800 MHz DDR2 SDRAM, Mac OS X Snow Leopard 10.6.6
- *OptiTrack* Motion Capture System [6] (nicknamed *MoCap*) made:
  - 12 IR Camera
  - Computer:
    - Software: MS Windows 7 Enterprise 64 bit and Arena 1.7 (data computing).
    - Hardware: Intel Core 2 DUO 3.00 GHz 4.00 GB RAM

Software

Brief introduction for presenting the utilized software.

- **Arena**: *MoCap* management program, installed in a computer with windows. It is used to set up the *MoCap*, manage skeletons and users, and save data.
- **OSCeleton**: simple network tool, interfaced with OpenNI and *Kinect*, streams data from the *Kinect* using Open Sound Control protocol over a UDP port.
- ***Max/MSP***: graphical development environment for music and multimedia, used to receive and save in a file all the data received form *OSCeleton*.
- ***MoCap* Toolbox**: is a *Matlab* toolbox, (dev. by University of Jyväskylä in Finland), that contains functions for the analysis and visualization of motion capture data.

3.1 Sampling data analysis

The resampling of the *MoCap* data is necessary for the comparison with the *Kinect* data. *MoCap* data are sampled in 100Hz, instead the *Kinect* data are in 30Hz. To compare them, the *MoCap* data was resampled from 100 to 30Hz, this transformation of the data could introduce errors and noise. *MoCap Toolbox* has an utility - *mcresample* - to change the sample of a *MoCap* file. A down sample from 100Hz to 30Hz implies a leak of data, but it is not know the quantity of noise that this operation adds to the data. *mcresample* utilizes a *Matlab* interpolation function - named *interp1* - that admits different methods, as reported in below list (citing from *Matlab* help):

- *nearest*: nearest neighbor interpolation
- *linear*: linear interpolation
- *spline*: piecewise cubic spline interpolation.
- *pchip*: shape-preserving piecewise cubic interpolation
- *cubic*: same as *pchip*
- *v5cubic*: the cubic interpolation from *Matlab* 5, which does not extrapolate and uses *spline* if X is not equally spaced.

All interpolation methods were tried to find the most fitted one. To determine what kind of interpolation is less noisy it was used the method depicted in the picture 3.1. First of all the *MoCap* data were sampled in 30 Hz, but the matrix data obtained weren't comparable anymore with the original, to fix this problem it was resampled back to 100Hz, in this way the distance between the two data structure could be calculated and the discrepancy could be seen. Two kind of analysis were performed with the *Distance* array (D). The first is called Signal to Noise ratio (SNR) and estimates how much the noise distorts the signal (in this case the signal is the information of the joints in the space) and the other one is the mean squared error. If S is the sampling function, T the sampled data $T = S^{-1}(S(M))$, $M[i,j]$ the original data matrix, the distance data D is the euclidean distance in 3D space:

$$D[i,j] =$$

$$\sqrt{(M[i,3j-2] - T[i,3j-2])^2 + (M[i,3j-1] - T[i,3j-1])^2 +}$$

$$\overline{+(M[i,3j] - T[i,3j])^2}$$

Matrix D has the same dimensions (rows and colums number) of the smallest matrix between M and T. The matrix is very huge, thus in appendix B are

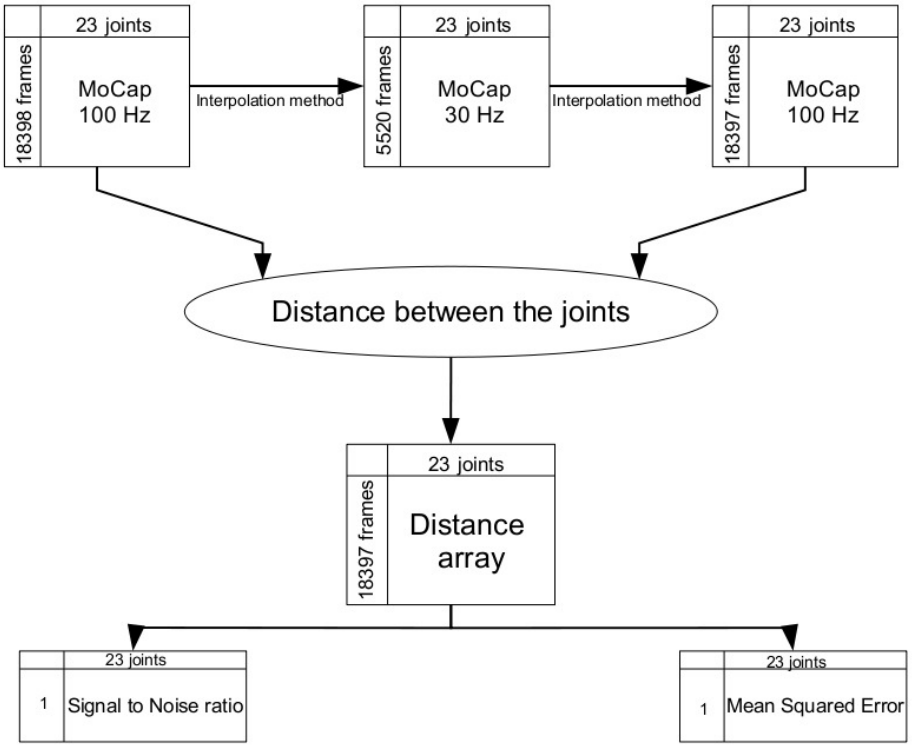


Figure 3.1: Graphical representation of the made calculations to verify the goodness of the sampling.

reported only the overall mean RMSE and the overall mean SNR for every joins and for every sampling method.

The figure 3.2 shows the *Boxplot* of the distance between the original and sampled joints, obtained with the *v5cubic* interpolation. The idea is: if the sampling does not introduce noise in the data, both RMSE and SNR only have values equal or close to zero (in the figure only a blue line in correspondence of the zero should appear). Obviously the sampling introduce some error, but as the figure shows the most part of values are close to zero (at least 75%), the red points are the outliers and they could be interpreted as the noise added by the sampling.

3.2 Synchronization analysis

As reported before there are two data structures, one output from *MoCap* and the other from *Kinect*, but it does not exist a synchronization mechanism between the two structures. To get a synchronization, the player at the beginning of the experiment performed some claps (in this case 3). The clapping was found

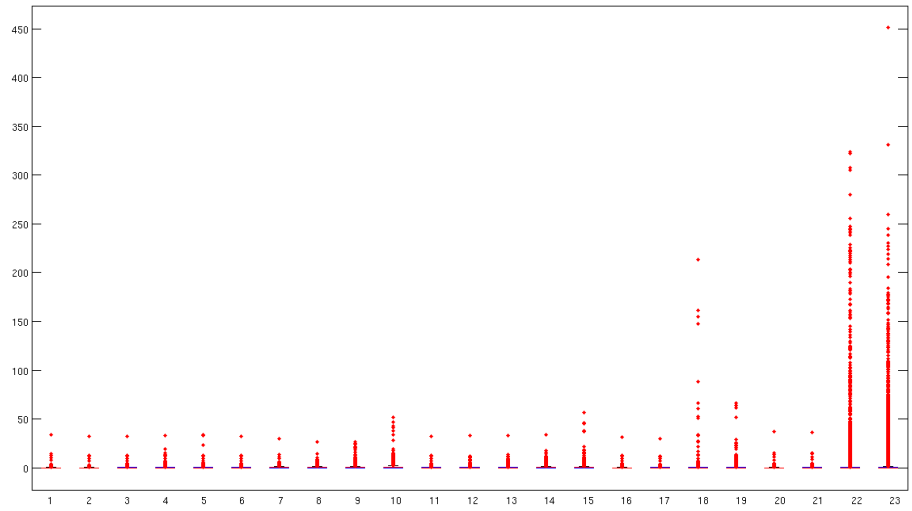


Figure 3.2: Boxplot of the error intruduced by the *v5cubic* interpolation.

in the recorded data as the frames where the distance between the two hands is minimized.

Before sampling the *MoCap* data, the clapping were searched. In table 3.1 is reported: the number of frame, where the distance between the hands is minimum (actually the distance between *Left* and *Right Wrist*, for the sake of stability), the minimum distance and the time interval between the claps, starting from the first clap.

Frame	Distance (mm)	Time interval (s)
4440	87.504	0.000
4527	81.216	0.870
4680	86.351	1.530

Table 3.1: Claps in the *MoCap*

The same calculation was made in *Kinect* data, in this case the *Hand* joints were used and it is reported in table 3.2.

Frame	Distance (mm)	Time interval (s)	Time interval First-Last (s)
291	120.206	0.000	
317	120.157	0.867	2.367*
362	123.256	1.500	

Table 3.2: Claps in *Kinect* data: (*) actually: $2.3\overline{6}$

The claps in the *MoCap* data, after the 100Hz to 30Hz sampling, is reported in table 3.3. There are some differences in the distance, but it has to take into

consideration the cubic interpolation performed by the sampling function as it had been already discussed in section 3.1.

Frame	Distance (mm)	Time interval (s)	Time interval First-Last (s)
1333	102.594	0.000	
1359	87.678	0.867	2.400
1405	114.645	1.533	

Table 3.3: Claps in *MoCap* data after the sampling.

The below formulas were used to transform the frame interval into the correspondent time interval, in detail, it is reported the formula for the 30Hz sampled data, i.e. the distance between the frames over the number of frame in a second:

$$\Delta t_0 = \frac{frame_0 - frame_{-1}}{30}$$

and the formula for the 100 Hz sampled data, with the same meaning of the previous one.

$$\Delta t_0 = \frac{frame_0 - frame_{-1}}{100}$$

Furthermore, the first clap was chosen as starting point (0.00 s).

3.3 Optimal roto-translation

After the synchronization of the data, the two collections of points were put in the same reference system. In *MoCap* a technician is able to use a calibration square to set the origin of axes in the room, instead using the *Kinect* it was impossible to set the reference center with something similar. As a matter of fact, *Kinect* uses its own sensor as origin of the axis, thus the reference system of the *Kinect* can not easily adapted to the *MoCap* one.

The first effort was to trace the position of the *Kinect* with 3 markers, in this way the position of the *Kinect* was obtained according to the *MoCap* system, but this method introduced parallax errors due to the positions of the markers that were placed in the external part of the cover. For that reason a mere translation was not sufficient to overcome the parallax errors so it was decided to use a more accurate rigid roto-translation without any scale or share.

To translate the *Kinect* reference system into the *MoCap* one, it was chosen to find the optimal roto-translation, a problem known as "Orthogonal Procrustes problem". Using the *Procrustes function* in *Matlab* the rotation matrix and the translation matrix were found, that minimize the mean squared error of the distance of the joints between the two skeletons. To calculate this matrix only

the joints listed in table 3.4 were taken into consideration, because the obtained results were better.

Name of *MoCap*'s joints	**Name of *Kinect*'s joints**
Right Ankle	Right Foot
Left Ankle	Left Foot
Chest	Torso
Site (Head)	Head
Left Shoulder	Left Shoulder
Right Shoulder	Right Shoulder
Right Hip	Right Hip
Left Hip	Left Hip
Right Elbow	Right Elbow
Left Elbow	Left Elbow

Table 3.4: Two set of points used as input in the *Procruste Problem*.

At the beginning, the initial frames (1-80) were assumed to be been sufficient to find an optimal solution of the *Procrustes problem*, because in that frames the player was performing a T-pose; but during the calculation it was realized that dropping all the other frame led a sub optimal solution. Consequently it was preferred to make this kind of calculation for every frame (1-4000). Precisely, for every frame a roto translation matrix was calculated, the transformation was applied to *Kinect* data, and then the mean squared error between the two skeletons was calculated taking into consideration all the 4000 frames.
At this point 4000 roto-translation matrices were obtained, that one with the least mean squared error was chosen and it was applied definitively to the *Kinect* data.

At the end *MoCap* and *Kinect* data were synchronized and put in the same reference system: the pictures 3.3 and 3.4 show the data sets before and after the optimal roto-translation.

Figure 3.3: Set of points before the roto translation (bird's-eye view).

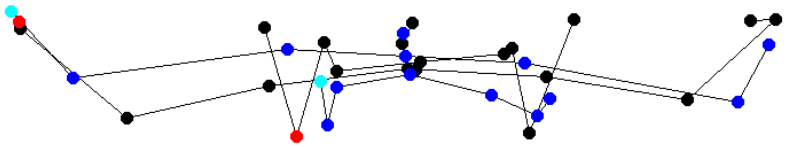


Figure 3.4: Sets of points after the optimal roto translation (bird's-eye view).

Chapter 4

Kinect data analysis

Before discussing about the error or better the noise of the *Kinect*, it is needed a summary.

1. The raw output data of the *Kinect* were translated in a *MoCap* data structure, compatible with the *MoCap Toolbox*.
2. The data output by the *MoCap System* were translated, using *Arena Software*, into a BVH file, this file contains the information about the joints, instead of C3D file which contains information about the markers.
3. The *MoCap* System data were re-sampled to 30Hz, to adapt to the *Kinect* one.
4. Both *MoCap* and *Kinect* data were synchronized.
5. *Kinect* data were roto-translated, to adapt to the *MoCap* reference systems
6. To calculate the distance between the *MoCap* and the *Kinect* data, they were merged into a common file.

Unlikely, all this transformations were insufficient to have a perfect match between *MoCap* and *Kinect* joints. The picture 4.1 puts in evidence all comparable joints with a red circle. Ideally, in case of perfect match, the *MoCap* joints and the *Kinect* ones would be superimposed, but here a difference is observable to the naked eye between the two skeleton systems. It means the two systems use different skeleton definitions.

The presence of two skeleton system implies a systematic error on all the measure. That means the error, as the distance between a *MoCap* joint and the equivalent *Kinect* one, cannot be calculated right now, but before the systematic error must be removed from the data. The systematic error for each joint is quantified calculating the average and the standard deviation of the x, y, z components of the vector that links the two comparable articulations during

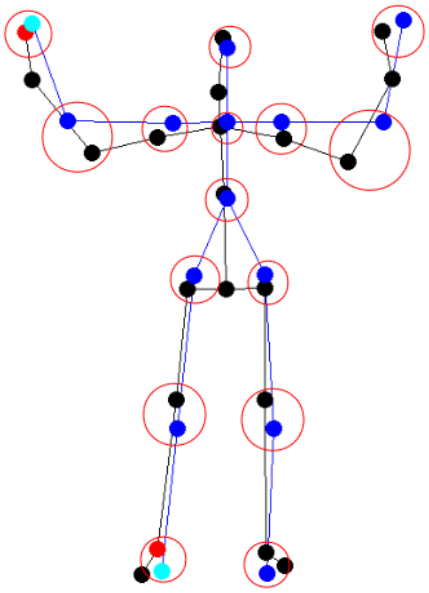


Figure 4.1: Comparison between the two skeleton definitions: the *MoCap* version is in black, while the *Kinect* one in blue, the red and the cyan joints are highlighted to show the right side of the skeletons.

the T-Pose (see fig. 4.2a). This vector is applied (as translation) to the *MoCap* joints before calculating the distance, in order to remove the systematic error (see fig. 4.2b).
Since the calculation of this systematic error is prone to error, when the vector distance between the *Kinect* joint and the translated *MoCap* joint was calculated, it was checked if its norm was greater or lower than the norm of the x, y, z components of the standard deviation: in the first case this value was classified as noise, in the second case it was classified inside the tolerance limits (see fig. 4.2c). The picture 4.2 summarizes these transformations.

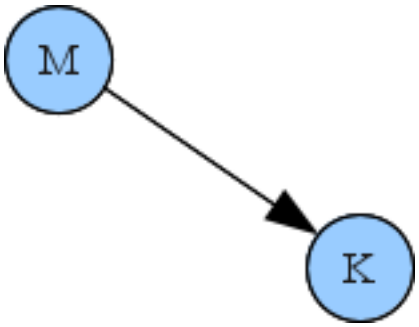
4.1 Results

The result is the real error that the *Kinect* device commits to track a human person. In tables 4.1 and 4.2 the Root Mean Squared Error (RMSE) of the *Kinect* is reported, defined as the average distance between *MoCap* and *Kinect* skeleton joints without the systematic errors. For every joint (in the column) was calculated the RMSE in mm in different situation (as reported in the rows). Finally the overall average error of every row and every column was calculated. The result was that the *Kinect* has an average error of 40.73mm, that is the *Kinect* is not more accurate than 4cm.

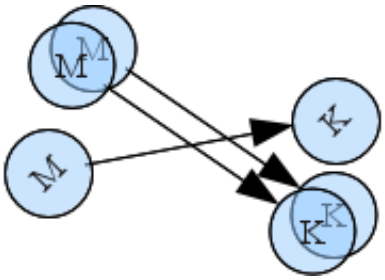
The last sentence could be evaluated as an hazard without discussing the real error of the *MoCap*. But the latter system has been already used and well known by IPEM researchers and its reliability is more than satisfactory for the research purpose and for the experiments conducted in the laboratory. For this reason the *MoCap* has been selected as the system of reference.

Accuracy of the *Kinect* is 4cm, that means the difference between *Kinect* and *MoCap* data in the localization of a joint is on average of 4cm, allows to understand the kind and the quality of movements of a player, the position of a person in relation to another, but not the quantity of a movement. For example use the *Kinect* to assess a piano player is not possible, nevertheless it is a really good instrument for entertainment and teaching applications, it is not good enough for quantitative research.

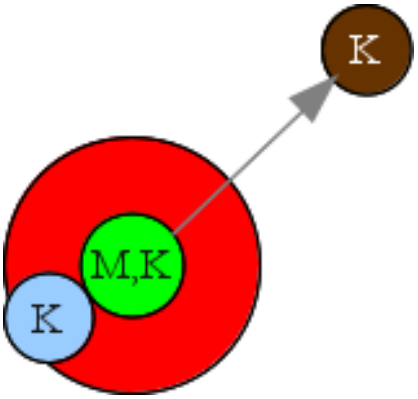
Finally, some boxplot examples are reported in figures 4.3 and 4.4, to understand the type of the error of the *Kinect* device. Boxplots represent a random type error, that it is mostly similar to a noise.



(a) Suppose, as example, M and K are the positions of a comparable joint, respectively of *MoCap* (M) and *Kinect* (K), in a frame of the T-Pose. The arrow depicts the distance vector between the two points.



(b) Frame over frame, this points are prone to noise. The average and the standard deviation of the component of each vector was taken. (T-Pose).



(c) The green point (M,K) is the ideal position of the joint without any error. The blue and the brown ones are example of real position of the *Kinect* points, in two consecutive frames. The blue one is inside the tolerance limits, instead of the brown one and thus the latter was classified as noise (represented by the gray arrow).

Figure 4.2: Graphical representation of the transformations to systematic error suppression

RMSE in mm	Head	Neck	Right Shoulder	Left Shoulder	Right Elbow	Left Elbow	Left Hand	Right Hand
1 - 80 T-Pose	8.62	14.25	20.10	10.49	49.75	71.74	30.45	14.66
275 - 400 Claps	15.73	16.59	31.67	13.50	51.21	57.26	42.95	50.58
400 - 550 Walk	14.96	17.52	25.05	23.82	56.25	65.79	57.29	57.29
638 - 770 First (Slow) Twirl	21.93	25.06	65.59	57.83	88.96	87.07	71.94	47.71
770 - 818 Second (Fast) Twirl	28.01	40.32	62.51	90.46	114.76	125.57	60.66	52.89
2250 - 2550 Some steps	27.84	35.66	44.20	52.12	84.81	63.94	91.58	71.59
2600 - 2757 Third (Very Slow) Twirl	24.91	34.27	77.58	66.27	107.31	132.28	35.31	41.14
2860 - 3085 Fast Hands movements	25.89	14.41	19.20	20.63	40.96	45.37	63.94	45.53
3150 - 4050 Free Movements	41.47	44.02	58.50	50.18	65.70	64.14	77.48	60.56
Average	23.26	26.90	44.94	42.81	73.30	79.24	59.07	49.11

Table 4.1: (Part 1) The distance (RMSE) of all the comparable joints of the *Kinect* data, assuming the *MoCap* data as exact. In coulmns are reported every Skeleton joints, in rows are reported the different actions performed by the player. All values are expressed in millimeters.

RMSE in mm	Chest	Right Hip	Left Hip	Right Knee	Left Knee	Right Ankle	Left Ankle	Average
1 - 80 T-Pose	13.18	16.41	25.94	9.09	31.21	15.41	25.27	23.77
275 - 400 Claps	11.54	10.19	5.95	2.87	5.22	10.19	5.97	22.10
400 - 550 Walk	20.93	29.43	28.09	15.90	7.81	14.37	9.90	29.63
638 - 770 First (Slow) Twirl	23.28	29.65	62.00	74.95	26.95	45.04	36.39	50.96
770 - 818 Second (Fast) Twirl	30.81	30.71	44.45	65.30	43.99	50.39	48.26	59.27
2250 - 2550 Some steps	38.40	30.84	29.45	43.46	38.54	53.48	40.14	49.74
2600 - 2757 Third (Very Slow) Twirl	29.68	29.06	51.99	62.81	38.65	45.08	34.76	54.07
2860 - 3085 Fast Hands movements	13.08	14.37	15.15	11.89	16.30	23.91	10.48	25.41
3150 - 4050 Free Movements	39.02	34.10	38.97	44.23	42.65	58.30	54.90	51.61
Average	24.44	24.97	33.55	36.72	27.93	35.13	29.56	40.73

Table 4.2: (Part 2) The distance (RMSE) of all the comparable joints of the *Kinect* data, assuming the *MoCap* data as exact. In columns are reported every Skeleton joints, in rows are reported the different actions performed by the player. The last value, down on the right is the overall average error, that is the error that the *Kinect* does in a joint tracking. All values are expressed in millimeters.

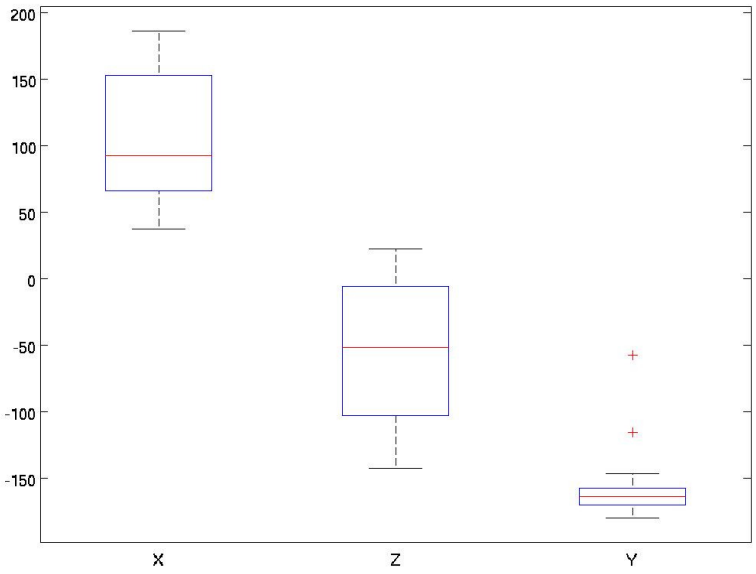


Figure 4.3: Noise of "Left Elbow", during the second twirl. The noise is separated in the x, y and z axis, the value are in mm.

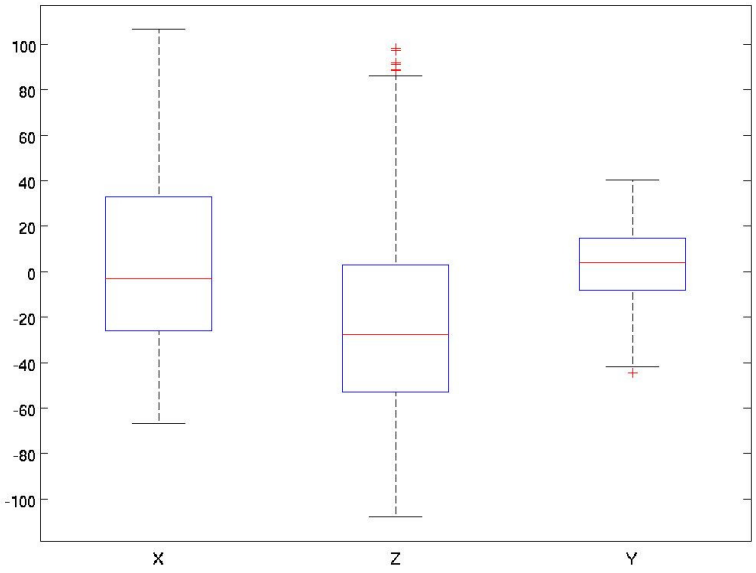


Figure 4.4: Noise of *Neck*, during free movements. The noise is separated in the x, y and z axis, the value are in mm.

Chapter 5

Delay analysis

The *Kinect* device analyzes its field of view, finds a human shape, if it is present, and returns the joint coordinates of the skeleton related to the human shape. Some operations are done by the *Kinect* hardware (VGA and depth recognition), others are carried out by the software (skeleton recognition). Moreover, drivers are responsible for the communication between hardware and software and the recognition of the device by operative system.

It is simple to see that the process which outputs the data introduces a delay: the purpose of this experiment is to estimate the delay of the *Kinect* in comparison with the *MoCap* system.

5.1 First experiment

5.1.1 Method

A *Max/MSP* patch was used in order to synchronize data streams recorded by *MoCap* and the *Kinect*. The patch was able to capture the data stream from *OSCeleton*, to save the data stream from *Arena* (*MoCap* software), and to record the input of the computer microphone.

To determine the delay of the *Kinect* the timing of hand claps was studied: the player performed some hand claps, while the *MoCap* system, the *Kinect*, and the microphone were recording together. All these recordings were stored into separate files, but they were already synchronized to each other thanks to the *Max/MSP* patch.

The experiment took place inside the *MoCap* area, the player was dressed in a special suit with markers, in order to be recognized by the *MoCap* system, but only the hands were captured. Data were sent by the *Arena* software to

a computer using *OSCNatNetClient*[1], they were translated to simple text by the *Max/MSP* patch and imported in *Matlab* using a custom developed parser. The parser reads the simple text file, then it organizes the information in a structured format readable by *Matlab*.

Similarly the *Kinect* data sent by *OSCeleton* were translated in a *Matlab* format, too. In this data was collected the full skeleton description of the player, but for the calculations only the information about the hand joints will be taken into consideration.

The hands of the player were located between 2.20 m and 2.50 m from the *Kinect*.

5.1.2 Data analysis

Clap analysis

To calculate the delay of the *Kinect* a player was recorded while he was doing some claps. A clap was identified in the data as the frame where the distance between the hand joints is minimal.
The same representation of the first column of the delay is depicted in illustration 5.1.

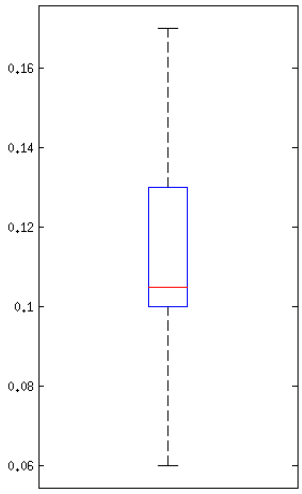


Figure 5.1: Boxplot of the delay of the claps. It is a graphical representation of the *Kinect* Delay (s) column of the table 5.1.

The column of the distance in the table 5.1 tells that the *Kinect* is not always able to track totally the movement of a clap: for example during the 9^{th} clap the *Kinect* stops tracking when the hands are 20 cm from each others. The *Kinect*

[1]OSCNetNatClient is a software application, developed at IPEM. It allows to stream *Arena* data to any computer connected through network, using OSC protocol.

usually stops tracking the hands before the real clap. To search the claps in the data, an algorithm that finds the frame where the distance between the hands is minimal was developed, but if the *Kinect* stops tracking before the real clap we obtain an advance clap.
Another weird phenomenon occurred between the frame 50 and 70. Here, the software swaps the right hand with the left one during a clap. Probably this phenomenon was due to a bug of the skeleton recognition software. The bug is shown in the figure 5.2, the figure represents the function on frame to the distance between the hands.

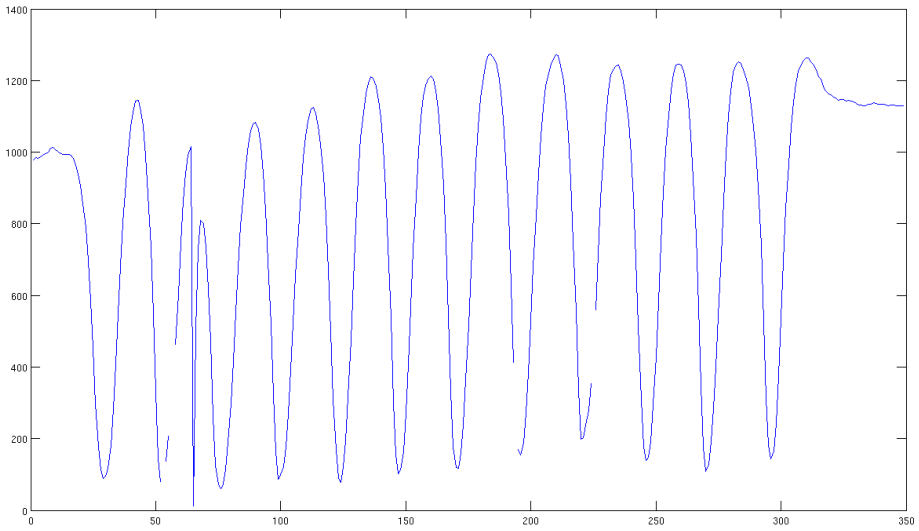


Figure 5.2: Distance between hands in function to frames. The reader can see around the 55^{th} frame an error, probably caused by *NITE* software.

The time in seconds can be retrieved starting from the frame number, indeed the *MoCap* system records a frame every $1/100^{\text{th}}$ seconds, instead of *Kinect* that records a frame every $1/30^{\text{th}}$ seconds. Now it is very simple to transform the frame number in seconds, applying the below formulas.

- For *Kinect* data (Sampled in 30 Hz):

$$t = \frac{n_{frame}}{30} \left(\frac{[frame]}{[frame/s]} = [s] \right)$$

- For *MoCap* data (Sampled in 100 Hz):

$$t = \frac{n_{frame}}{100} \left(\frac{[frame]}{[frame/s]} = [s] \right)$$

Kinect			*MoCap*			*Kinect*	Audio Peaks		*MoCap*
Frame	**Dist. (mm)**	**Time (s)**	**Frame**	**Dist. (mm)**	**Time (s)**	**Delay (s)**	**Sample**	**Time(s)**	**Delay (s)**
29	88.93	0.97	84	75.34	0.84	0.13	36815	0.83	0.01
54	137.45	1.80	163	88.79	1.63	0.17	71815	1.63	0.00
76	61.12	2.53	241	85.54	2.41	0.12	105937	2.40	0.01
99	86.19	3.30	320	82.35	3.20	0.10	141027	3.20	0.00
124	77.50	4.13	400	82.22	4.00	0.13	176325	4.00	0.00
147	102.22	4.90	480	81.24	4.80	0.10	211592	4.80	0.00
171	116.36	5.70	557	95.43	5.57	0.13	245689	5.57	0.00
196	154.57	6.53	642	93.06	6.42	0.11	282775	6.41	0.01
220	197.90	7.33	727	71.74	7.27	0.06	320469	7.27	0.00
246	140.03	8.20	810	89.40	8.10	0.10	357246	8.10	0.00
270	109.37	9.00	891	75.41	8.91	0.09	392886	8.91	0.00
296	143.83	9.87	977	72.41	9.77	0.10	430765	9.77	0.00
					Average	0.11±0.03			0.00±0.00

Table 5.1: The table is made by three sub-tables (*Kinect* *MoCap* and Audio Peaks), in *Kinect* and *MoCap* sub-tables every row lists the frame of the clap, minimal distance between the hands and the correspondent time. The column, on the right of the *MoCap* sub-table, is the delay, calculated as difference between the value of the Time column of *Kinect* and the *MoCap* ones. The last row is the average of all the values of the columns of the delays. Estimated values are depicted in red. In the sub-table named audio peaks there are the samples at which, in the audio data, a peak of the noise of a clap is found; the reader can note looking at the Delay column on the right that the timing of the audio is strictly compatible with the timing of *MoCap*: the velocity of the *MoCap* system is the same as the velocity of the sound.

Maximum distance analysis

The problems encountered analyzing the claps suggests that this is not the best method to study the delay of the *Kinect*. It is simple to notice that every clap is constituted by a moment when the hands are at the minimum distance and a opposite moment when the hands are far from each other (at the maximum distance). The distance between the hands is a function with points of local maximum (openings) as well as local minimum (claps). Now the points where the hands are the farthest from each other will be studied: in the figure 5.3 is depicted graphically the calculated delay reported in table 5.2.

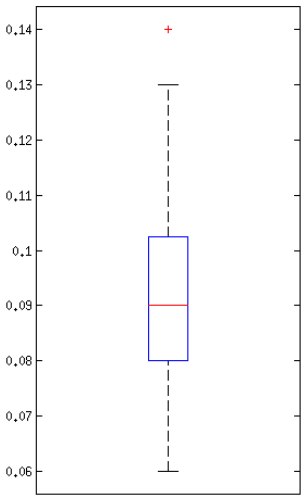


Figure 5.3: Representation of the *Kinect* delay (s) column as presented in table 5.2.

After finding the local maximum points, the lapsed time was obtained easily dividing the frame number with the frame rate, a chronological moment was obtained when the hands were farthest from each other. The difference between the *Kinect* and the *MoCap* time gives a not constant delay, probably it depends on the software and on how the software is handled by the operative systems.

The reaction time of an individual is the time required for an observer to respond to the presence of a stimulus [36]. The value of the delay of the *Kinect* is as good as the reaction time of an individual to respond to a visual stimulus (180-200 ms) or an auditory stimulus (140-160 ms).

However if we consider the time lapsed in order to a stimulus reaches the brain, the *Kinect* results, more or less, ten times slower than the time elapsed for an auditory stimulus (8-10 ms) and up to five times slower than the time elapsed for an visual stimulus (20-40 ms). The information about the reaction time and the time necessary for a stimulus to reach the brain is in [37].

Hands tracking analysis.

Difficulties in the claps tracking have suggested to do an experiment to better understand this phenomenon. These experiments are complementary and not strictly related to the thesis work, because they were conducted without support of the *MoCap* system, so it is impossible to make a comparison. Here it is presented an analysis of the claps performed in different velocities.

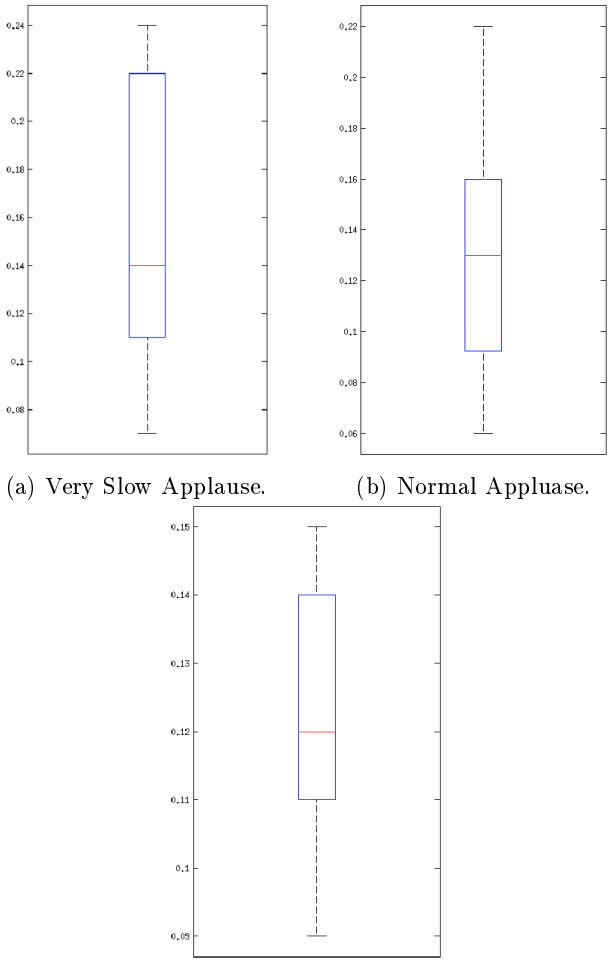
Purpose, materials and method

To understand how the *Kinect* recognizes clap movements, only the *Kinect* and the Mac Desktop were used, with the software listed previously. Furthermore a microphone was used to have a time reference: recording the clap sound, the timestamps were obtained between different claps checking the peaks in the *wave* file. The experiment consisted in a player performing some applause in different velocities, while the instrumentations were capturing him.

Data analysis and results

Handling the data found out that *NITE* has some problem tracking claps movements. Even if, randomly or without any apparent reason, some joints of the arms disappear during movements, it is possible to develop a software in order to recognize all the present claps in the *Kinect* data.

Basically there is not any correlation between the velocity of the clap and the accuracy of the *Kinect*, as the reader can see reading the results (table 5.3, figure 5.4), in fact the average values do not change in correspondence of the velocities. For the same reason the values of the delay are more or less the same, taking into consideration that the skeleton recognition is managed by *NITE* software and its velocities are dependent by the load of the operative system.



(a) Very Slow Applause. (b) Normal Appluase.

(c) Very Fast Applause

Figure 5.4: Graphical representation of the delay in relation to the velocities of the claps.

Kinect			*MoCap*			*Kinect*
Frame	**Dist. (mm)**	**Time (s)**	**Frame**	**Dist. (mm)**	**Time (s)**	**Delay (s)**
15	993.83	0.50	40	887.07	0.40	0.10
43	1143.90	1.43	133	1016.41	1.33	0.10
64	1016.45	2.13	207	923.26	2.07	0.06
90	1082.71	3.00	289	958.88	2.89	0.11
113	1125.60	3.77	368	995.32	3.68	0.09
136	1210.17	4.53	446	1063.81	4.46	0.07
160	1211.67	5.33	523	1068.68	5.23	0.10
184	1273.06	6.13	605	1104.06	6.05	0.08
210	1272.65	7.00	691	1094.86	6.91	0.09
235	1243.43	7.83	770	1077.67	7.70	0.13
259	1244.66	8.63	855	1093.68	8.55	0.08
283	1251.52	9.43	934	1095.75	9.34	0.09
310	1263.92	10.33	1019	1098.81	10.19	0.14
					Average	0.10±0.02

Table 5.2: Opening hands analysis. Table is made by two sub-tables (*Kinect* and *MoCap*), in whose every row lists the frame, the distance between the hands and the time where the hands are at the maximum distance each other. The last column on the right is the delay, calculated as difference between the value of the Time column of *Kinect* and the *MoCap* ones. The last row is the average of all the values of the column of the delay. The estimated values are depicted in red.

Experiment:	**Minimum Distance (m)**	**Maximum Distance (m)**	**Delay (s)**
Very Slow Applause	2.29	2.97	0.15±0.06
Normal Applause	2.15	2.54	0.13±0.04
Very Fast Applause	2.23	2.46	0.12±0.02

Table 5.3: The average distance between the hands and the *Kinect* sensor in the 3 experiments, in the last column the average delay. See appendix A for having more details.

5.2 Second experiment

5.2.1 Claps analysis

The data collected only with *Kinect* experiments suggests to re do the experiment with both *MoCap* and *Kinect* systems. In table 5.4 is reported the distance in the clap, that is better than in previous experiment. But in the same time, some problem during the tracking of the clap (when the two hands are close each other) were detected: for example missed frames problem.

5.2.2 Openings analysis

The problems found in the tracking of clap-movements suggest to discuss other analysis techniques: one of these is the analysis of the opening of the arms during a clap-movement. This kind of movement is the opposite of the clap and the experiments seemed to show that it is more recognizable by the *NITE* software. In table 5.5 we can see the opening movements analysis, with the average delay of the *Kinect*.

5.3 Summing up

Done Experiments confirmed the *Kinect* has a delay of about 0.1 seconds, but it is not clear if the delay depends only on hardware device, more probably it depends both on hardware and software. Unfortunately these experiments are not able to state how much delay is introduced by hardware or by software.

From specifics the *Kinect* produce a new image data every 33 ms, assuming trivial the delay introduced by the USB cable, the total amount of delay added by the software, that is operative system and *NITE* middleware, is about 77 ms[2].

[2] That is 100ms (experimental delay) minus 33 ms

Kinect			*MoCap*			*Kinect*	Audio Peaks		*MoCap*
Frame	**Dist.(mm)**	**Time(s)**	**Frame**	**Dist.(mm)**	**Time(s)**	**Delay(s)**	**Sample**	**Time(s)**	**Delay(s)**
186	42.02	6.20	602	110.78	6.02	0.18	265621	6.02	0.00
239	70.05	7.97	779	119.50	7.79	0.18	343874	7.80	-0.01
280	67.93	9.33	920	123.53	9.20	0.13	405683	9.20	0.00
322	44.19	10.73	1058	107.27	10.58	0.15	465207	10.55	0.03
359	55.79	11.97	1181	111.52	11.81	0.16	521084	11.82	-0.01
396	72.18	13.20	1306	119.13	13.06	0.14	576158	13.06	0.00
434	97.40	14.47	1438	124.31	14.38	0.09	634709	14.39	-0.01
475	67.84	15.83	1568	118.99	15.68	0.15	691603	15.68	0.00
511	85.57	17.03	1693	116.68	16.93	0.10	747072	16.94	-0.01
553	61.59	18.43	1828	93.30	18.28	0.15	805850	18.27	0.01
599	114.72	19.97	1962	116.43	19.62	0.35	865701	19.63	-0.01
634	66.12	21.13	2098	118.24	20.98	0.15	925709	20.99	-0.01
674	62.17	22.47	2232	121.82	22.32	0.15	984883	22.33	-0.01
720	25.60	24.00	2373	117.43	23.73	0.27	1047100	23.74	-0.01
758	53.69	25.27	2512	113.73	25.12	0.15	1108247	25.13	-0.01
800	54.93	26.67	2645	123.75	26.45	0.22	1166761	26.46	-0.01
837	75.17	27.90	2777	96.57	27.77	0.13	1224916	27.78	-0.01
876	77.38	29.20	2911	92.77	29.11	0.09	1283678	29.11	0.00
921	17.36	30.70	3044	107.78	30.44	0.26	1342959	30.45	-0.01
958	71.17	31.93	3186	105.68	31.86	0.07	1403680	31.83	0.03
1001	62.01	33.37	3324	99.94	33.24	0.13	1463425	33.18	0.06
Average:	64±22			112±10		0.16±0.07			0.00±0.02

Table 5.4: The **clap analysis** table is made by three sub-tables (*Kinect*, *MoCap* and Audio Peak), in *Kinect* and *MoCap* sub-tables every row lists the frame of the clap, minimal distance between the hands and the correspondent time. The column, on the right of the *MoCap* sub-table, is the delay, calculated as difference between the value of the Time column of *Kinect* and the *MoCap* ones. Last row is the average of all the values of the columns of the delays. In Audio Peaks are listed the samples, at which the clap was occurred. In the last column we can see that the *MoCap* system does not introduce delay.

Kinect			*MoCap*			*Kinect*
Frame	**Distance (mm)**	**Time (s)**	**Frame**	**Distance (mm)**	**Time (s)**	**Delay (s)**
171	1084.96	5.70	559	1001.59	5.59	0.11
212	1260.60	7.07	694	1116.03	6.94	0.13
264	1191.75	8.80	868	1057.49	8.68	0.12
302	1148.21	10.07	1000	1016.50	10.00	0.07
342	1137.55	11.40	1132	1002.53	11.32	0.08
381	1150.52	12.70	1258	1017.48	12.58	0.12
420	1183.02	14.00	1394	1053.77	13.94	0.06
456	1212.06	15.20	1516	1069.03	15.16	0.04
495	1130.61	16.50	1644	1005.89	16.44	0.06
537	1168.35	17.90	1782	1044.68	17.82	0.08
576	1180.36	19.20	1912	1039.24	19.12	0.08
618	1148.30	20.60	2050	1020.86	20.50	0.10
659	1091.94	21.97	2189	1000.38	21.89	0.08
699	1188.84	23.30	2323	1052.72	23.23	0.07
740	1154.01	24.67	2463	1038.57	24.63	0.04
782	1154.41	26.07	2599	1027.79	25.99	0.08
821	1174.37	27.37	2728	1026.67	27.28	0.09
860	1225.08	28.67	2862	1067.25	28.62	0.05
901	1217.23	30.03	2998	1052.04	29.98	0.05
942	1213.10	31.40	3135	1057.55	31.35	0.05
984	1164.81	32.80	3275	1033.84	32.75	0.05
					Average	0.08±0.03

Table 5.5: Opening hands analysis. The table is made by two sub-tables (*Kinect* and *MoCap*), in whose every row lists the frame, the distance between the hands and the time where the hands are at the maximum distance each other. The last column on the right is the delay, calculated as difference between the value of the Time column of *Kinect* and the *MoCap* ones. The last row is the average of all the values of the column of the delay.

Chapter 6

Different gestures analysis

In executed experiments some problems have experienced, regarding to clap recognition by *Kinect* device and its software. The purpose of this experiment is to understand the limits of the *Kinect* in the recognition of a simple movement "a clap".

6.1 Method

A player wearing *MoCap* gloves executed 3 claps every time, in different positions, looking the *Kinect* with different angles. Figure 6.1 shows a representation of the experiment, the little squares represent the positions of the person when he was doing the claps a roman number inside each square is the order of execution of the applauses. For every position the direction of the player is depicted using an arrow and two eyes. The External circle is the field of the *MoCap* system, the *Kinect* device was putted near the border, in order to maximize the examination area, furthermore 3 markers were leaned on it, to have a feedback of its position by the *MoCap*.

6.2 Results

Let start discussing with the best position (I), the player was at the optimal distance from the sensor of *Kinect* because he was centered and at the minimal distance in which all the joints were captured and were visible by the sensor. As a matter of fact no problem was reported.

Right and left position (II-III): the player moved first to his right, later to his left. In the data there were some missing frames, when the hands were closing each other in the act of the clap. The player was not looking to the sensor, his shoulders made a 0 degree angle with the X axis, so for example when the

Claps frame intervals	Δx (mm)		Δz (mm)		Distance from *Kinect* (mm)
	Right	Left	Right	Left	
I 137 - 199	423	-244	2177	2184	2235
II 458 - 511	1133	536	2334	2329	2513
III 670 - 728	-275	-926	2260	2294	2404
IV 875 - 934	382	-254	1437	1484	1526
V 1079 -1135	329	-260	3292	3292	3324
VI 1567 -1609	1015	603	2162	2338	2422
VII 1696-1744	-357	-831	2280	2085	2302

Table 6.1: In first column there is frame intervals when the tree claps are executed, in the second and third column there is the average of the value of distance (mm) taken from the sensor, factorized in X and Z (lateral and depth), and divided by left and right hand. The last is the average euclidean distance of the hands from the *Kinect*. All this values are taken with the *MoCap* system, taking as reference point the *Kinect*. (Measures are expressed in mm).

Claps frame intervals	Δx (mm)		Δz (mm)		Distance from *Kinect* (mm)
	Right	Left	Right	Left	
I 137 - 199	440	-267	2104	2121	2163
II 458 - 511	1200	545	2270	2306	2480
III 670 - 728	-276	-978	2238	2251	2362
IV 875 - 934	NaN	NaN	NaN	NaN	NaN
V 1079 -1135	361	-384	3287	3317	3343
VI 1567 -1609	993	603	2257	2139	2362
VII 1696-1744	-351	-804	2022	2239	2238

Table 6.2: Distance of the hands, taken from *Kinect* sensor. The subtraction of the values of this above table and the values taken from *MoCap* is the error of the *Kinect*. The average of this error is 42mm, compatible with the overall error found in the "Free movements experiment": see 4.1. The 4^{th} row is constituted only by *Not a Number* values, indeed the claps movements were completely missed in the data.

player was on the right, the right arm was partially covered by the left side of the body and harder to track by the sensor. A similar thing occurred when the player was on the left.

When the player was in the IV position, his legs and his face were not visible by the sensor, as result *NITE* lost the skeleton's user, *OSCeleton* stopped to send data and the *Max/MSP* patch interrupted to write data in the log file. On the other hand the *MoCap* was still sending data to the patch and the patch was recording them in another file. As final result there were 146 missed frame from the *Kinect* log file. However the information recorded by the *MoCap* system was sufficient to fix the problem, adding NaN values instead of *Kinect* missed frames. Concluding the player must be more than 1.5m far from the *Kinect*. Further experiments found out that it is sufficient to have the upper part of the body, like arms, head and torso, to be recognized by the *NITE* software.

The fifth (V) position represented the other limit of the *Kinect* the maximum distance. Despite the fact that *Kinect* lost a lot of frames, the clap-movements were recognized by the software.

Finally the (VI-VII) positions were the same of the second and third ones, but in this case the user was looking at the sensor and the tracking of the *Kinect* worked better. A reason could be that the *Kinect* was able to distinguish both the arms and so *NITE* software gathered more information than in II and III positions.

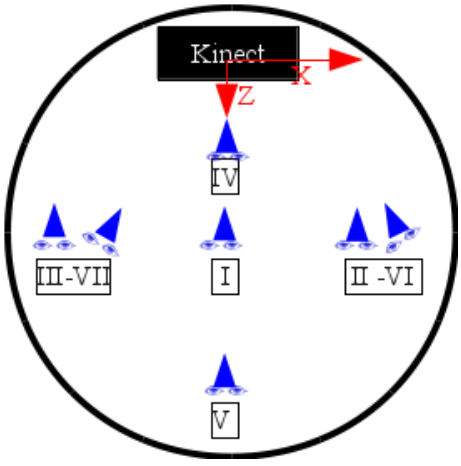


Figure 6.1: Graphical representation of the experiment, positions of the different claps are specified in roman number following the order of execution.

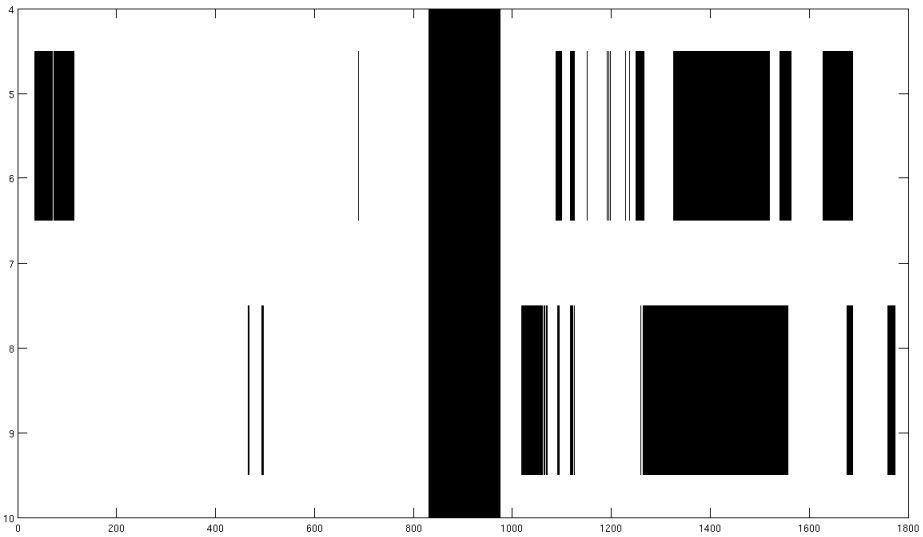


Figure 6.2: *Kinect* Data: Missing frame analysis. List of joints: 5 – right elbow, 6 – right hand, 8 – left elbow and 9 – right hand. The vertical black line between 800 and 1000 describes the lack of data when the *Kinect* software stopped itself to track the user, after it lost him.

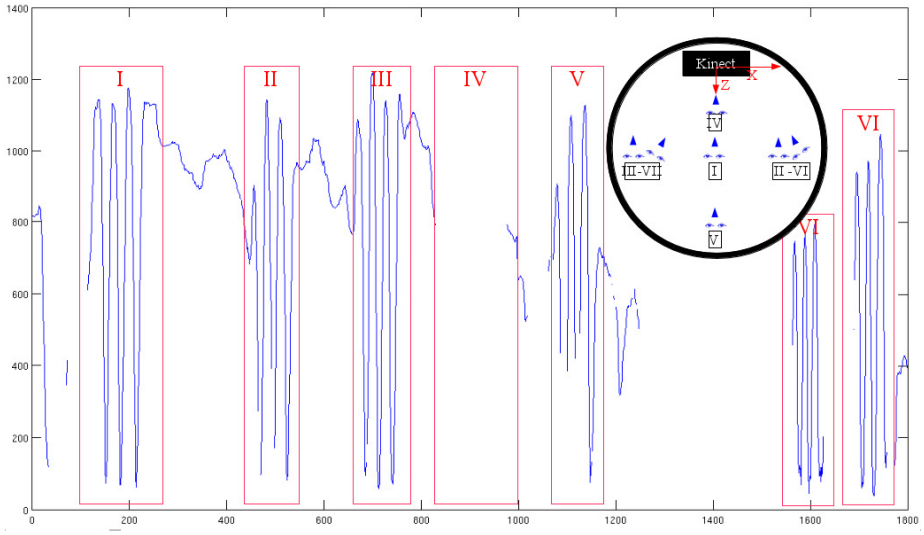


Figure 6.3: Distance Hands function, *Kinect*. The numbered red squares depict the claps, the number corresponds to a order and respectly to a position, how reported in the circle.

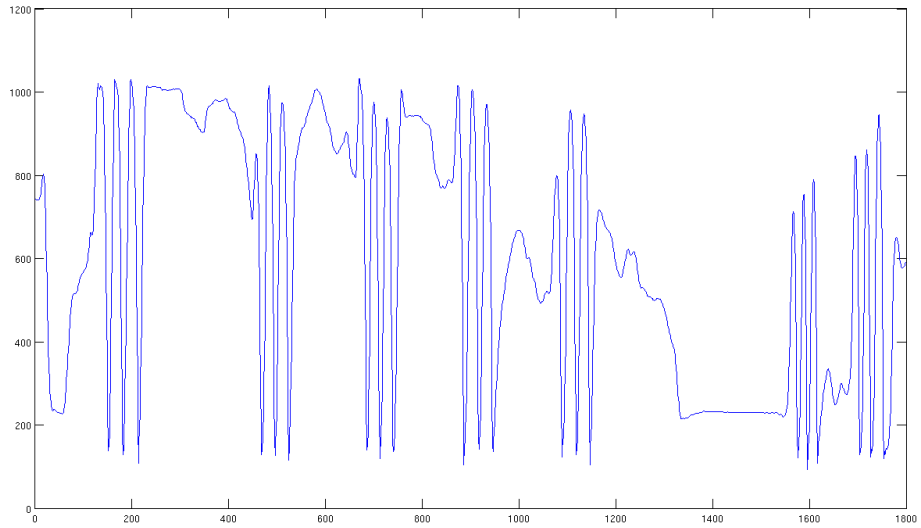


Figure 6.4: Hand Distance Function, *MoCap*. There is no discontinuity in data.

Chapter 7

Accuracy of *Kinect*

In the "Free Movements" experiment has been already analyzed the goodness of the *Kinect* in relation to different kind of movements, but all this movements were made close to the optimal *Kinect* point of view. Found an average error in the accuracy of the analyzed device around 4cm, this experiment will verify the behavior of the error in relation to the distance of the user from the sensor (see section 4.1).

7.1 Method

The experiment is divided in 4 parts (see figure 7.1), in relation to the distance from the *Kinect* sensor, while the 5^{th} part is only as control and reference. The person started from about 2.5m far from *Kinect* and then he shifted even far of about 50cm every time, till the *Kinect* and the *MoCap* system were unable to track his position.

In every segment he tried to redo the same movements, even if for a human is impossible to make exactly the same movements, in order to minimize the error due to different movements.

He was always looking the *Kinect*, that means the player was, jumping up, left and right and moving his arms like a bird on the fly, without any twirl.

At the end of the experiment the user was still in a position of relax, with the arms and the legs gently open, looking at the sensor at the optimal distance of about 2.5m. In this manner it was possible to collected some data in optimal conditions, the values and the results could be useful as confrontation.

During the experiment the field was put in evidence with some object, in order that the user could recognize it and to avoid that he went out. The reader can have an idea of the area of work looking at the picture 7.1, right area in evidence with the green color.

Position of player	Min (m)	Max (m)	Average (m)
I segment	2.542	2.845	2.636
II segment	2.933	3.427	3.161
III segment	3.228	3.613	3.488
IV segment	3.702	4.118	3.973
V segment	2.583	2.639	2.607

Table 7.1: The max, min and average distance between the *Kinect* sensor and the player in every segment in which the experiment is divided. The value of the distance were calculated as the average of the distances of all the joints of the user's skeleton, frame over frame. This operation prints a function that map every frame in a distance point.

The physical space of the experiment was delimited by the operative range of the *MoCap* system and the limits of the *Kinect*, the picture 7.1 renders the area taken in examination.

The area of work of the *MoCap* system was marked with a white mat on the floor of the lab, shaped as a circle of 5.95m of diameter. Both *Kinect* and player were inside this mat, because in this way the *MoCap* could be used as reference systems of the distances.

However, as specified in [38], the *Kinect* has a depth range of work between 1.2 and 3.5m and an angular field of view of 57° horizontally and 43° vertically, that means a square frustum with, starting from the sensor, the closer base with size 1.3x0.95 m and the other with size 3.8x2.8m, that means an area of $6m^2$ and a volume of about $12m^3$. All these values are inside the range of work of the *MoCap* system without any problem.

The range of work would be closer to the sensor, but the *Kinect* was unable to track the whole skeleton of the user, when he was closer than 2.50m. That is the reason because the experiment work began at the distance of 2.50m instead of 1.2m (the theoretical minimal distance).

7.2 Data analysis and results

First of all let put in evidence the error of the *Kinect* to estimate the distance of the individual person from its sensor. The *MoCap* system was able to detect the *Kinect* because was defined a rigid body, locating three marker on the top of the object, in this way we had, all the time, the position of the *Kinect* in relation to the position of the user.

Summarizing we have, in the data, the position of the markers of the *Kinect* (3) in relation to all the joints that constitute a human skeleton; both the sets of data could be seen, frame by frame, as 'rigid body', hence, to calculate the distance between the two, the center of mass for each rigid body was calculated,

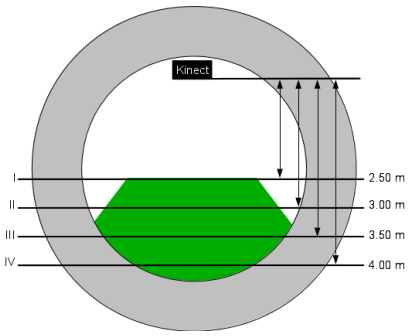


Figure 7.1: Range of work (in green), that is the intersection between the *MoCap* range of work and the *Kinect* one. The horrizontal line are the 4 distances in relation with the 4 parts of the experiment

and then the distance between the both centroids. The table 7.2 shows the difference between the distance calculated in the MoCap system and the *Kinect* system, using the procedure described previously. Let take in consideration the segments I, II and III, that is those where the player is inside the optimal range of work of the device, the accuracy on the depth estimation is about 10cm.

Segments:	**Average Error (mm)**	**Standard Deviation (mm)**
I (2.50 m)	60	71
II (3.00 m)	79	85
III (3.50 m)	149	90
IV (4.00 m)	33	78
V (2.5 m)	168	28

Table 7.2: Error in realtion to the distance of the player from the *Kinect*. Calculated as difference between the distance of the user from the *Kinect* taken by *Kinect* and the same distance taken by the *MoCap* System.

From average values there is no correlation between the distance and the error of the *Kinect*. Let consider this measures were taken into consideration the operative field of the Kinect. Meanwhile it confirmed the previously examination, that is the RMSE of the *Kinect* is around 4cm, see tables 7.3 and 7.4.

Part	Head	Neck	Right Shoulder	Left Shoulder	Right Elbow	Left Elbow	Left Hand	Right Hand
2.5 m (I)	18.86	21.28	21.89	26.64	86.54	71.93	51.44	46.83
3.0 m (II)	40.08	38.46	42.78	36.58	71.41	79.47	47.69	48.54
3.5 m (III)	17.10	22.76	34.53	22.04	58.70	65.57	68.24	51.57
4.0 m (IV)	28.11	29.86	24.46	42.51	119.32	52.62	40.60	75.13
Ref. (V)	16.37	16.14	21.81	19.13	88.10	29.26	18.98	48.75
Average	24.10	25.70	29.09	29.38	84.81	59.77	45.39	54.16

Table 7.3: (Part1) The table summarizes the Root Mean Squared Error of the position of the joints in the space, having as reference the *MoCap* coordinates.

Part	Chest	Right Hip	Left Hip	Right Knee	Left Knee	Right Ankle	Left Ankle	Average
2.5 m (I)	13.76	25.31	27.72	17.09	17.04	33.61	32.78	34.18
3.0 m (II)	29.81	34.80	34.92	21.98	19.69	30.83	28.02	40.34
3.5 m (III)	18.95	40.36	31.40	15.57	14.93	33.20	27.59	34.83
4.0 m (IV)	28.23	45.20	38.55	37.95	42.87	72.07	53.83	48.75
Ref. (V)	7.41	17.20	13.27	17.70	20.34	20.44	6.06	24.06
Average	19.63	32.57	29.17	22.06	22.97	38.03	29.65	36.43

Table 7.4: (Part2) The table summarizes the Root Mean Squared Error of the position of the joints in the space, having as reference the *MoCap* coordinates.

Chapter 8

SoundingARM

*SoundingARM*idea: the realization of a system to aid visually impaired people to explore quickly a familiar environment after a staying in hospital, or to explore a less familiar environment such as an hotel room, with the aim being simple, not intrusive and cheap.

A possible user is a patient who has sustained severe traumas in his visual system, causing him the totally or partially blindness. A patient has initially many difficulties: he has to change his life-stile and to learn how to compensate a so important leak. Studies consider the visual system as the primary source of information about the surrounding area [11], so an individual who is blind has to rely on other sensory channels to obtain appropriate spatial information regarding his surroundings [39]. Indeed, it is generally believed that an individual who is blind (both early and late onset) develop compensatory behavioral strategies through the use of his remaining senses [40, 41].

With respect to navigation, information captured through sound is very important for developing a sense of spatial orientation and distance as well as obstacle detection and avoidance [42, 43]. Previous work with individuals who are blind has shown that spatial information obtained through novel computer-based approaches using sound [44, 45] may prove useful for developing navigation skills. In parallel, many advances in computer technology have improved information accessibility in general. For example, many individuals with visual impairment are familiar with speech-based systems (e.g., screen readers or text to speech interfaces) as well as contextual nonspeech information (e.g., alerts using associative and realistic sounds) [46].

SoundingARM is previously installed by qualified staff in a patient's room, on an appropriate place, because the demo could not to be shifted anymore, after a installation in a room and consequent creation of the configuration file. When a patient open the entrance door, the *Kinect* device is able to detect his skeleton

and *SoundingARM* is able to track his arm and what he is pointing. If he points something *SoundingARM* emits a sound by the loudspeakers: currently the sound is only the name of the pointed object, future developments will introduce 3D audio spatialization and sonification in order to give to user a human friendly information about distances and features of pointed objects.

8.1 Hardware

Hardware components must meet Microsoft's *Kinect* sensor prerequisites (publicized in `www.kinectforwindows.com`). However in this implementation it is used a computer with the following setup:

- Monitor: 15.6".
- Processor: CORE I7-2670QM.
- Hard Disk 640GB.
- RAM 4GB.
- graphics card: AMD 6490M 1GB GDDR5.
- Windows 7 HOME (64bit).

Obviously, connected to this computer there are a *Kinect* sensor (with a USB cable) and two loudspeakers.

8.2 Software

Software prerequisites for *SoundingARM* are Microsoft *Kinect* SDK and *Pure Data*. The application was written in C++ using Microsoft Visual Studio 2010 with .NET framework 4.0, it implements UDP sockets able to communicate with programs using OSC protocol as *Pure Data*. Windows 7 has already installed a text to speech synthesizer, but by default only with an English voice, thus it was needed to install an Italian voice.

Pure Data is distributed with the patch *system* that permits to execute terminal commands. But in Windows operative systems, *system* launches a child terminal session that preempts *Pure Data* till its closure. Thus *system* cannot be used to launch a background server application. To overcome the problem another patch was developed - named *createprocess* - technically is a wrapper of *CreateProcess* function, defined in MSDN Windows API, briefly *CreateProcess* creates a new process and its primary thread. The new process runs in the security context of the calling process, avoiding preemption.

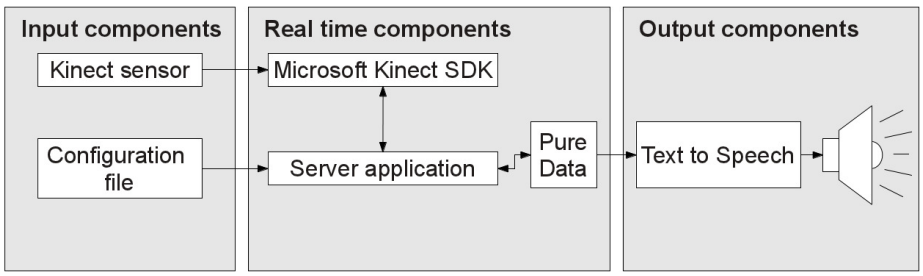


Figure 8.1: *SoundingARM* system architecture.

8.3 System architecture

The system architecture (fig. 8.1) of the project is made of:

- **Input component**: Microsoft *Kinect* sensor, connected by USB to a computer with Windows 7 and Microsoft *Kinect* SDK for the recognition of the skeleton. A configuration file, containing information about the furniture of the room: this file is loaded once by the real time application at starting.
- **Real-time processing components**: the server application that analyzes the user skeleton and understands what item the user is indicating, the *Pure Data* patch that works as an intermediary between the application and the text to speech server.
- **Multimedia output component**: text to speech server, using Microsoft Speech API engine, gives a feedback to the user of what he is indicating through stereo speakers.

The configuration file contains the information about the furniture of a room, for every object. For example a desk is defined as the parallelepiped that contains object itself, calculating four base points, the height of the object and the height of the object from the floor. The base points have to be taken considering the *Kinect* Sensor how reference system. In this file are reported all the relatively big and fixed objects of a room (for example desks, tables, wardrobe, couches) in order to give to the user a spatial idea, to discern tools from obstacle; all moving objects (for example chairs) are excluded and so are not mapped, in fact, the application is unable to manage changing or shifting of objects.

The kernel of the system is constituted by the sever application. When it starts, it parses the configuration file initializing a data structure that represent the room furniture, after that, the configuration file is closed and the data will remain the same during all the program execution. During the normal working it checks the presence of a human person, calculates the prospective plan of

view of the room that the user should have in front of him in function of his own head position and it checks if the user is indicating something. If the user is indicating a object the correspondent name (in raw text format) is sent by the server application to the *Pure Data* patch.

The *Pure Data* patch has a double feature: first of all it acts as a graphic user interface, to start and control the application, the other functionality is to interface the Application with the Text to Speech server, using OSC packet.

Software output component is composed by a Text to Speech synthesizer, it utilizes Microsoft Speech API that it is available in Windows 7, without any installation: an English voice is already installed by default, but not an Italian voice, necessary for the nationality of consumers. A simple way to install other voices is *eSpeak*, a text to speech application, that it can be used with Microsoft Speech API. Hardware components are two normal stereo speakers connected to PC audio output jack, they reproduce to the user the spoken version of the textual name of the indicated object.

8.4 Software development

8.4.1 Input components

In figure 8.2 is reported a fragment of a real configuration file, that contains information about an environment, specifying the positions of the objects in relation to the sensor. Configuration file is loaded by the user during the launch of the application. Obviously different rooms has different files, that means the user may try the exploration of his living room or his kitchen staying in a different place, for example when he is in an hospital. However the purpose of *SoundingARM* is aid visually impaired people a moment before they enter and begin to looking for something.

The file has to be got ready before starting the application and with a correct structure, if possible using UTF-8 encoding. Referring to the figure 8.3, the name of a object is holden between square brackets, for example "`[Desk]`", and characterizing attributes are listed under the name as a couple "`key = value;`". "`p1`", "`p2`", "`p3`", "`p4`" are four base point of a virtual parallelepiped that includes the edge desk, the values between brackets are the coordinates "`(x,z)`" from the sensor, for example the point "`p1`" is 2.8m on the left and 1.75m behind the sensor. Attribute "`heightOfObject`" is the height of the object, in this case the height of the desk. Instead, the "`heightFromFloor`" is the height of the object from the floor, in this case its value is 0, that means the desk is leaned over the floor. But for example "`heightFromFloor`" is different than 0 are objects leaned on the wall: cupboards, televisions, and switches. The "`[Kinect]`" item is mandatory, it specifies the height from the floor of the sensor, this value must be most accurate as possible, because it is utilized in further calculation.

The name between square brackets has actually another important function: it is utilized by text-to-speech synthesizer to "say" the name of the pointed object. Theoretically there are no limitations in string length, however let consider the time spent by the text-to-speech synthesizer to play the text. A trivial experiment was made to test the quickness of *SoundingARM*, substituting the names of the objects with harmonic frequencies, obtaining a very simple sonification of the room. Latter example is meaningful of the modularity of the application, in fact, this simple change was made with a custom *Pure Data* patch, without altering the application itself. After that it was the first experiment for future developments of a 3D audio spatialization, for example.

A fragment of an input file

```
[Kinect]
height = 660;

[Desk]
p1 = (-2800, -1750);
p2 = (-2000, -1750);
p3 = (-2800, 50);
p4 = (-2000, 50);
heightOfObject = 730;
heightFromFloor = 0;
```

Figure 8.2: Supposing there is a desk in a room, after placed the sensor, the installer has to measure the distance of the desk from the sensor and fill all the required data. The height of the *Kinect* is the height of the sensor from the floor.

Figure 8.3: The mapping of a desk: the four feet form a rectangular base, that includes all edges of the desk. The parallelepiped that include the desk is defined multiplying the base with the desk height.

8.5 Real Time components

SoundingARM server application receives the input file with information about a composition of a room. That information is parsed and stored in RAM using a custom data structure, constituted by classes `Room` and `Furniture`. A room is constituted by a set of pieces of furniture, that is a `Room` object is an array of `Furniture`, while a `Furniture` object represents the attributes of a piece of furniture (for example see [Desk] in figure 8.2). In addition `Room` class store the height of the *Kinect* sensor and other useful variables.

8.5.1 Virtual Room creation

After the parsing of configuration file *SoundingARM* has the positions of all the mapped objects of a room, when a user is detected, to understand what he is pointing, the application must make additional computation to simulate the "view" of the user. The main idea is create a projection plane mapping in a bi-dimensional array the angular view that the user has of a object. The class that build this projection plane is called `ProjectionSphere`, in fact, the plane could be thought as a sphere that surrounds the user's head with radius the extension of his arm. To have a right room representation the application has to calculate the solid angle that subtends the object. In order to simplify, the solid angle is factorized in its vertical and horizontal components.

8.5.2 Calculation of projection user's view

Considering the desk in figure 8.4, the server application have to find the solid angle α that subtends an object for a user (H). First the application calculates the parallelepiped that includes the object as a set of 8 points: four lower base points (t_i) and other four upper points (t_{u_i}). To have details the most meaningful steps are mathematically explained and reported. Remeber vertex K (*Kinect*) is the reference system, all the other coordinates are expressed in function of K: the application knows the position of P_1 from configuration file, and the position of the user thanks to the *Kinect* sensor.

t_i	*i-th* lower base point.
t_{u_i}	*i-th* upper base point.
k_h	*Kinect* height.
h_f	height of the object from the floor.
h	height of the object.
p_{i_x}	x component of ith base point
p_{i_z}	z component of ith base point

Considering p_i points reported in configuration file, the position of the parallelepiped that includes the object is found.

$$\begin{aligned} t_i &= (p_{i_x}, h_f - k_h, p_{i_z}) \quad \text{for } i = 1, \ldots, 4 \\ t_{u_i} &= (p_{i_x}, h_f - k_h + h, p_{i_z}) \quad \text{for } i = 1, \ldots, 4 \end{aligned}$$

Thus the calculation of the solid angle is performed, in this thesis is reported only the horizontal estimate of the solid angular, because the vertical one is quite similar. Be $\vec{v}$ the vector from user's head to a point of the base of the object, and (u_{p_x}, u_{p_y}, u_{p_z}) user's position.

$$\begin{aligned} \vec{v} &= (v_x, v_y, v_z) \\ t_i &= (t_{i_x}, t_{i_y}, t_{i_z}) \\ v_x &= t_{i_x} - u_{p_x} \\ v_y &= t_{i_y} - u_{p_y} \\ v_x &= t_{i_z} - u_{p_z} \end{aligned}$$

Be user (H), *Kinect* (K), and base point (P_1) vertexes of triangle KHP_1 (see fig. 8.4), the problem is find the angle $K\widehat{H}P_1$ knowing the length of its edges: $\overline{PK}$, $\overline{PH}$, and $\overline{KH}$.

$$\begin{aligned} \overline{PK} &= \sqrt{t_{i_x}^2 + t_{i_z}^2} \\ \overline{PH} &= \sqrt{v_x^2 + t_z^2} \\ \overline{KH} &= \sqrt{u_{p_x}^2 + u_{p_z}^2} \end{aligned}$$

Composing Herone's formula, for the calculation of the surface of a triangle, and law of sines, the angle $K\widehat{H}P_1$ (β in fig. 8.4) is got by:

$$\begin{aligned} K\widehat{H}P_1 &= \left\lceil \arcsin\left(\tfrac{2S}{\overline{KH}\cdot\overline{PH}}\right) \cdot \tfrac{180}{\pi} \right\rceil \\ &\text{where,} \\ p &= \tfrac{\overline{PK}\cdot\overline{PH}\cdot\overline{KH}}{2} \text{ and} \\ S &= \sqrt{p\left(p - \overline{PK}\right)\left(p - \overline{PH}\right)\left(p - \overline{KH}\right)} \end{aligned}$$

These calculations are repeated for all eight points that constitutes the parallelepiped, and at the end it is chosen the maximal angle that is possible obtain subtracting pairwise every angle. In figure 8.4 α is obtained subtracting γ to β, but it is not the maximal achievable angle, in fact α' is the right angular user's view, obtained subtracting $K\widehat{H}P_3$ to β.

The same procedure is followed to calculate the vertical angular component, with some differences for example it is considered different pairs of points. As in the previous case, the aim is to estimate δ (see fig. 8.5), knowing the coordinates of the lower and upper base point. The calculation uses the same trigonometric

calculation before explained, but instead of *Kinect* sensor are utilized the user's feet. Note that the application does not need to know the real position of the user's feet, in fact, the magnitude of angles δ' and δ'' is independent of user's feet point F, it is enough to take a fictional point perpendicular to the floor and having the same x and z coordinates of the user's head. The angle $F\widehat{H}P_i$ is calculated for every point of parallelepiped that includes the object, (for example in the figure 8.5, the angles δ' and δ''), and only in a second moment is founded the best fit angle in relation to the user's view (δ in figure).

8.5.3 Creation of the `ProjectionSphere`

After all the angles are computed, a object is represented as two pairs [$\alpha_{x_{min}}$, $\alpha_{x_{max}}$], and [$\alpha_{y_{min}}$, $\alpha_{y_{max}}$], respectively horizontal and vertical bounds of angular user's view of the considered object. The values spaces between 0 and 180 (degree) and are integer. They can be used to fill an bi-dimensional array (180x180) of integer that represents the `ProjectionSphere`, i.e. the data object that simulates the user's view of the room. The indexes of the array corresponding to the real angular view of the user, from left to right (in rows from 0 to 180) and from downwards to upwards (in column from 0 to 180).

Assuming definitions of objects, in the configuration file, as a sequence, the application assigns an increasing cardinal number in relation to the order in file. In this case, with only an object, the application will assign 0 to the Desk, this number will become an identifier (ID) for the object. Every object defined in configuration file has an number, and it is utilized in the data structure. For example calling the bi-dimensional array `projectionSphere`, assuming "Desk" (ID=0) have a solid angle bounded horizontally by [75°, 105°], and vertically by [45°, 70°] (figure 8.6), the operation of filling are reported in figure 8.7. `projectionSphere` is initialized with value -1, and only in a second moment is fulfilled with the objects, at the end the matrix has values greater or equal than zero only where a object is defined.

8.5.4 Running of *SoundingARM*

Now the reader is ready to understand what happens during a normal running of *SoundingARM*. After installation of the demo in an environment and consequently creation of configuration file, the application is ready to start and to look for a user. If the user is found, the *Kinect* sensor gets his head position, and the calculation of the `ProjectionSphere` begins (see 8.5.2). The calculation of `ProjectionSphere` is CPU-intensive, moreover the user could move himself continuously. To reduce CPU time only "head movements" greater than 10cm are considered. That means the user can move and walking around, but

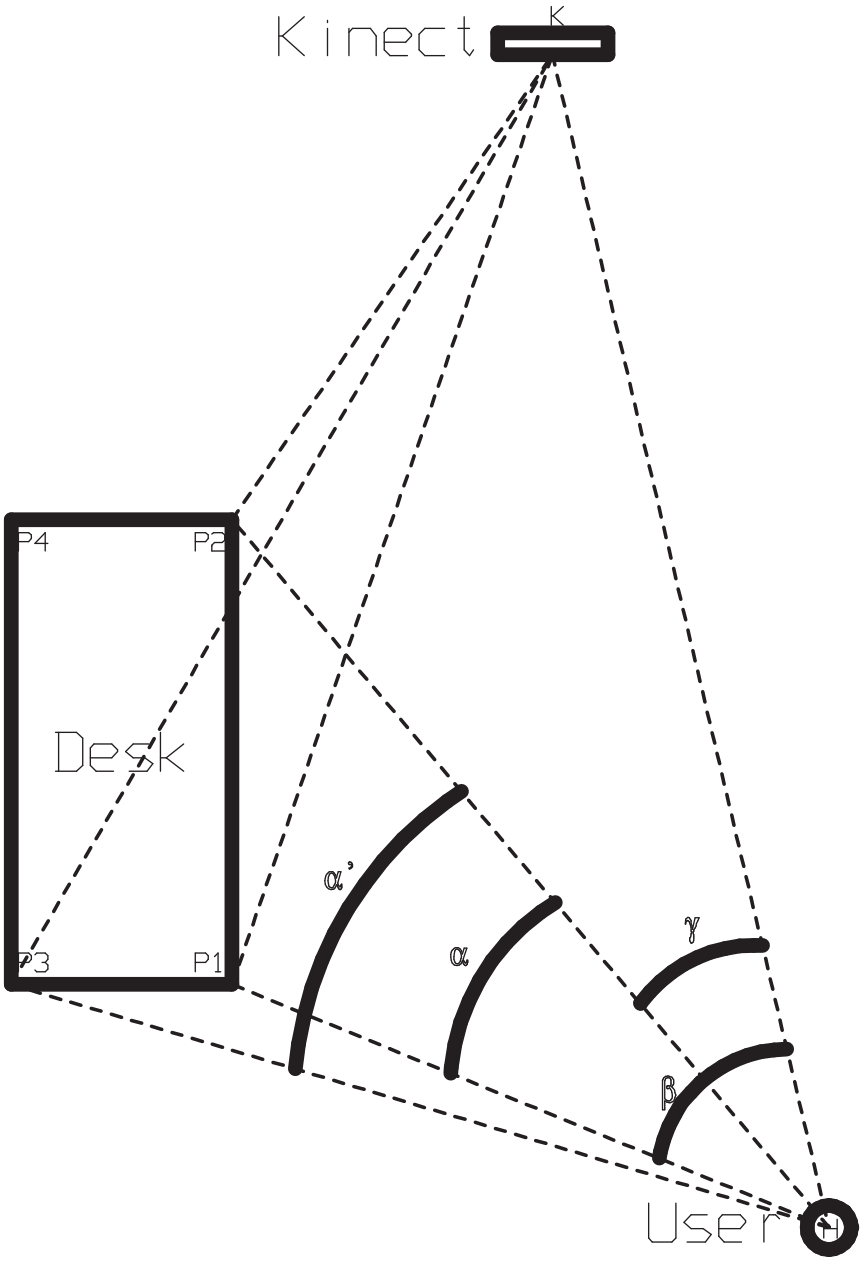


Figure 8.4: Schematic representation of angular calculation of the horizontal plane (bird's-eye view).

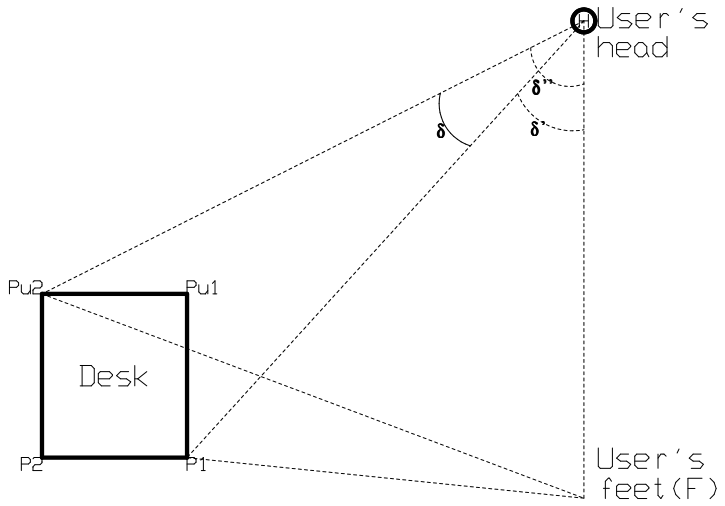


Figure 8.5: Schematic representation of angular calculation of the vertical plane (lateral view).

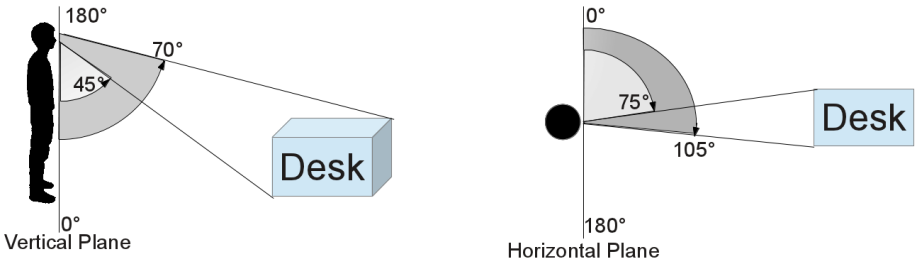


Figure 8.6: A person has a desk in front of him: side view representation on the right and bird's eye view representation on the left. He sees this desk under an angle, whose magnitude is in function of his position in respect to desk position. This angle can be factorized in a horizontal and in a vertical component. The Vertical Plane and the Horizontal Plane are imaginary projective planes, defining the orientation of the user view.

Insertion "Desk" in data structure.

```
for i = 75 → 105 do
    for i = 45 → 70 do
        proiectionSphere[i, j] ← 0          ▷ ID of desk
    end for
end for
```

Figure 8.7: `proiectionSphere` is a bi-dimensional matrix representing the user's virtual view. Matrix is fulfilled assuming "Desk" (ID=0) have a solid angle bounded horizontally by [75°, 105°], and vertically by [45°, 70°].

recalculation of his virtual projection view does not occur for movements lesser than 10cm. This measure is not casual, in fact, it is a good trade-off between the accuracy of the sensor and its mean depth error in user's position estimation.

During a normal execution, the application try to get information every 300ms about user's head and his right or left arm. Let be the configuration as in figure 8.6, H the user's position and M hand position. *SoundingARM* calculates the magnitude of the solid angle that makes $\vec{HM}$, vector that joins user's head with hand. Vector $\vec{HM}$ has two angular components: the horizontal one, whose magnitude varies from the left to the right of a user with his view to *Kinect* sensor, and the vertical one, whose magnitude varies form down to up. As soon as both components of the angle are known, these are used as index to have access to `projectionSphere` array, that returns the ID of the pointed object: if the user is indicating something or -1 otherwise.

SoundingARM does not distinguish right from left hand, people can change their arms when they want. Often a user move both the hands, while is pointing with ones, he makes short movements with the other. Obviously the application has to detect whose arm tracking. For this reason a cylindric shadow area is defined for 35cm around the user, his hands have to exceed the shadow area to be tracked by *SoundingARM*.

8.6 *Pure Data* patch

SoundingARM server application works similar to a background application, it runs without any control interface, but it was programmed to wait for commands sent using OSC over UDP[1] port 7660. *Pure Data* patch provides a user friendly control interface to manage *SoundingARM* (see figure 8.9), in details it authorizes starting the application and the text to speech synthesizer. Start-

[1] Open Sound Control over Datagram protocol: *SoundingARM* server application can accept network packet sent with OSC standard.

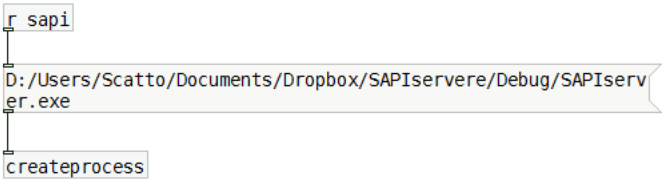


Figure 8.8: Example of use of createprocess patch. The input message must contains a path of a Windows executable file.

ing and stopping of skeleton recognition and eventually the killing of the server application are specific commands sent by a OSC format packet to the server application itself.

To start executable files (like *SoundingARM* server component) the patch uses *createprocess* (figure 8.8), a specific developed external that permits to execute a Windows 7 application in background simulating a double click in an execution file without preemption and unsightly opening of terminal windows. *createprocess* substitutes the *system* patch, latter is already present in a normal *Pure Data* distribution, but it is affected by a preemption problem in Windows: in other words *system* patch is unable to start a background process, and blocks *Pure Data* during its execution rendering unusable the real time computation. Technically *createprocess* is a *Pure Data* patch that wraps the homonymous Windows function, whose creates a new process and its primary thread, as specified in Microsoft Developer Network. The new process runs in the security context of the calling process, isolated and independent from *Pure Data* DSP engine.

SoundingARM server application transmits all the useful data to the patch, using always OSC packets, the output interface is reported in figure 8.10. Data contains information about the user's and user's hands position, position of a pointed object, and its name. This part of patch is not yet utilized, however it allows future development of, for example, a 3D audio spatialization. The OSC standard does not preclude the use of other applications similar or different to *Pure Data* and gives incoming developers the choice; however *Pure Data* was chosen for being vastly known in CSC[2] department.

8.7 Output components

At the present time the patch forwards the name of a selected object to a text to speech synthesizer. It is a simple application that wait on port 5000 and as soon as it receive a string, recalls the Microsoft Speech synthesizer, already

[2]Centro di Sonologia Computazionale - University of Padova.

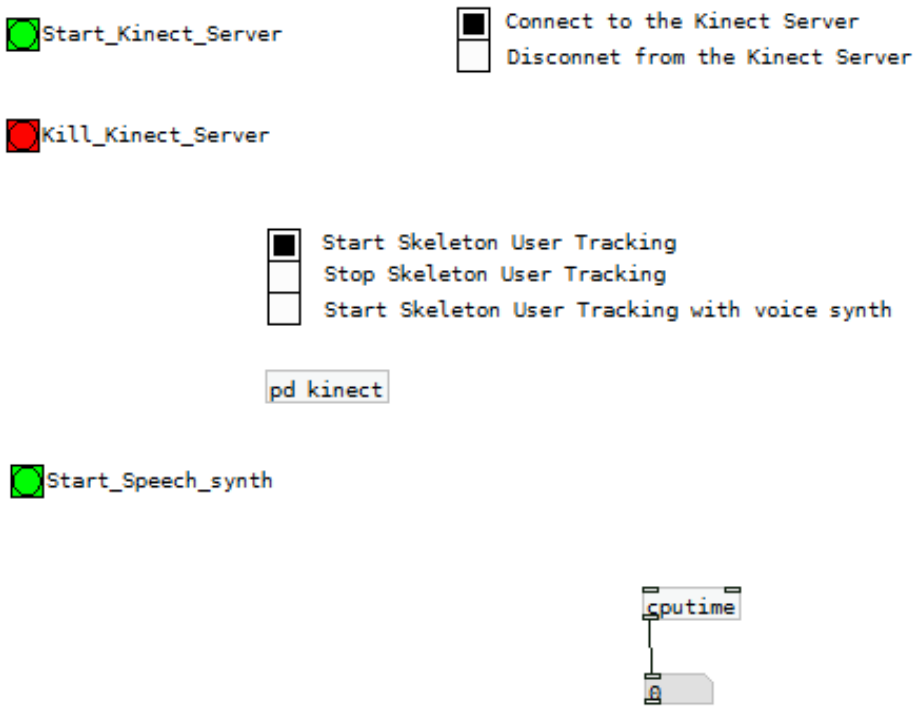


Figure 8.9: Control panel of *SoundingARM Pure Data* patch

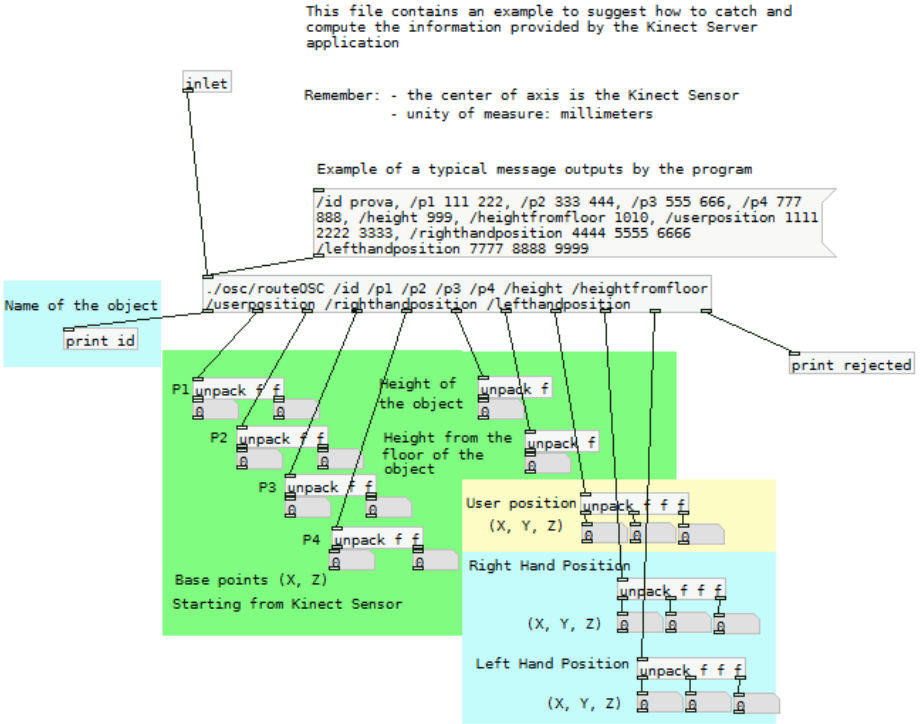


Figure 8.10: Output panel of *SoundingARM Pure Data* patch

installed in Windows 7. The reproducing of a human voice is demanded to the SAPI[3] engine. Default voice is English, the English output of Italian words was unrealistic and creepy to listen: fortunately installation of an Italian female voice fixed the problem.

Nevertheless text to speech is a good source of information for visually impaired people, a synthetic voice could be felt cold and uncomfortable. A solution could be the addition of a 3D audio rendering using audio signals, and auditory icons obtaining a demo more similar to an auditory visual. Any solution will be chosen it is not problematic, in fact, *SoundingARM* server application and control patch were created from the beginning to be adapted for future developments.

8.8 Preliminary testing

The overall purpose of the preliminary testing were to ensure if the *SoundingARM* system performs at an acceptable level for the user. It is necessary, indeed, that the system must first be tested by the users in order to ensure that they have a positive and efficient experience in using it. *SoundingARM*, on March 6, 2012, has been installed in a kitchen of "Istituto Regionale Rittmeyer per i Ciechi" (in Trieste, Italy) 8.11, and tested with ten users with different degrees of visual disability and with cognitive impairment associated. Identified testing participants received instructions prior to the start of preliminary testing. Almost all of them identified the objects in the kitchen and enjoyed the experience but we observed that these users need of many verbal explanations and a "physical guide" in the first approach with the system, a tactile signal on the floor in order to maintain the position on the doorway (they tend to enter the room as usual they do). In the preliminary testing, *SoundingARM* was used also by blind people in wheelchair; in these cases the system had difficulties in providing the user with all the acoustic feedback: the wheelchairs constitute an obstacle for the recognition of the skeleton.

In addition, the users with cognitive impairment had to be guided by an external operator whose body interfered with the user's skeleton recognition, leading *SoundingARM* to make errors. This problem can be fixed adding a multi-user management. The finger pointing gesture has to be as direct as possible.

Generally, blind people never localize any target (acoustic or not) by pointing to it: when trying to reach a sonorous object, a blind person usually gropes for it rather than reaching for it directly [47]. Consequently, the users, especially

[3]Microsoft Speech API development kit for audio synthesis and recognition.

those who have cognitive impairments associated, need for a brief period of training, in order to become a more accurate as possible [48].

Figure 8.11: Example frames of a video registered in the Kitchen of "Istituto Regionale Rittmeyer per i Ciechi" (in Trieste, Italy). Dott. S. Zanolla was testing *SoundingARM* before the experimental phase with patients.

Chapter 9

Conclusions

Main objectives of this thesis were the analyses of different motion capture systems, the definition, and the development of *SoundingARM*: a *Kinect* application to aid visually impaired people in exploration of a domestic environment or an unknown hotel room, for example. Analysis of different motion capture systems was done at University of Gent, at Institute for Psychoacoustics and Electronic Music (IPEM), where a very accurate and rather expensive device - called *MoCap* - is installed for tracking people's movements and behaviors. After, the thesis continued at University of Padova, at Centro di Sonologia Computazionale, at Department of Information Engineering, with the making up of the system called *SoundingARM*.

In this final chapter the obtained results will be discussed, both from the study of accuracy of the *Kinect* and from the development of *SoundingARM*. Before some impressions will presented about the device from Microsoft, built during the done experience in the realization of this thesis.

A pros of the *Kinect* device is very simple to use. It is born as a console game device, hence for a user the only thing to do is plug it in the Xbox 360, that means it does not need technical skill for the installation, just follow the instructions in the user guide.

The *Kinect* can be installed and connected to a computer, using OpenNI or Microsoft *Kinect* SDK. All the softwares are available in Internet and it is possible to put them in a bundle version, in respect to relative license restrictions. For example *NITE* (PrimeSense drivers for *Kinect*) is not an open source project, it is developed and owned by PrimeSense, but it is under a license that allows the software development for free and commercial use. In other words, in a future software development, it is possible to bundle all this software in an installation file, in order to, for the final client, the installation of the Kinect looks like a game installation.

The OpenNI solution does not implement the audio yet, that is it is impossible to use the microphones of the Kinect, that is a considerable limitation. Neither in the future the things will change, probably, because the *Kinect* was assembled by Rare, utilizing different hardware components among which the PrimeSense processor. PrimeSense uses its knowledge about its own Sensor to develop the OpenNI project in competition with Microsoft. But then the PrimeSense seems to recognize officially another kind of 3D sensor, the X-tion Pro developed by ASUS [49], while the compatibility with the Microsoft *Kinect* is thanks to a mod developed by "avin2".

Kinect is well commercialized in all the world, and it begins to be well known, in fact, also thanks to thumping advertisements [50], Microsoft was able to sell more than 10 million devices in more or less five months [51, 52]. The Microsoft SDK was released to help the world of research and the enthusiastic communities in the developing of their application, it is compatible only with Windows 7 and its license does not allow the commercial use [53].

This thesis work has tried to understand the excellence of the device, during the experiments it was found out that the noise in the data is very high, hence the use of a good filter should be necessary in a final release of an application. Filters were not utilized during the evaluation phase, because the purpose was to evaluate the excellence of the device and not the filter one. All measurements were made taking the average and the standard deviation in relation to *MoCap*, in order to remove white noises and obtain the error that the Kinect makes locating a point in the space.

Summarizing the error found is more or less 4cm in the direction X and Y, that are respectively width and height, and around 10cm in the Z (depth) axis. These accuracies allows to understand the kind and the quality of movements of a player, the position of a person in relation to another, but not the quantity of a movement, for example use the *Kinect* to assess a piano player is not possible. It is very good for entertainment and teaching applications but not good enough for quantitative research. The delay of the device is about 0.1 second, that means it allows developing of application with a feedback video.

The Kinect and its software tracks good simple and empathized movements, but the problems begin when the sensor cannot detect all the joints, when the user is not looking the *Kinect*, when he is not at the right distance from the sensor or when he is doing strange positions. This kind of problems should be taken into consideration during the develop of a *Kinect* application. For example arm tracking when the arms are to close to the torso, leg tracking when the legs are crossed or close each other, user tracking when the user is too close to a wall or partially occluded by an object and very fast movements in general. Some of this issues could be fixed in the future releases of *NITE* or

Microsoft *Kinect* SDK, but for now the only thing to do is compensate them writing robust applications.

The evaluation of *Kinect* simplified the creation of *SoundingARM*, a *Kinect* application for aiding visually impaired people. In fact, the information collected was enough to anticipate issues like estimate errors, defining specific filter, and shortening the overall development time. Instead, the choice of Microsoft *Kinect* SDK was quite forced, to use *Kinect* tilt engine, and microphones. Tilt engine is used at the launch of *SoundingARM* to locate the sensor parallel to the floor plane, while microphones could be potentially utilized in the future.

Researchers of CSC[1] had the possibility to install *SoundingARM* in a kitchen of the "Istituto Regionale Rittmeyer per i Ciechi" (in Trieste, Italy). Preliminary results are positive, the application match overall the patient type, whom it was created. The test phase involved ten users with different degrees of visual disability and with cognitive impairment associated, the application works well with visually impaired patient, instead it had some difficulties with patient both blind and with cognitive impairment. The latter patients needed a tutor to help them to move and pointing object, but the application were not developed for multi-user situations, thus some problem of skeleton clones occurred. Another type of problem arose with patients in wheelchair, in some case the skeleton recognition for a user could be problematic. It is a well known issue, by the Microsoft *Kinect* SDK developer, too.

> "Smaller wheelchairs, such as hospital-style chairs or racing chairs, seem to work best with Kinect. Also, large protruding armrests (such as the control arm on some motorized wheelchairs) may inhibit recognition by Kinect as the sensor might recognize them as another set of arms. Try to move these arms out of the sensor's view."

Cited from: `http://support.xbox.com`

SoundingARM is still a prototype application, it needs further experimental tests, probably at Rittmeyer, that is glad to host experiments. In addition, some software adjustment have to be done to make the application more stable in case of multi-user situations, or in case of problematic patients, like a patient in wheelchair. Future developments of *SoundingARM* will be able to account more efficacy. Some improvements could be, for example, smarten up the GUI, introduce a multi-user management, and a sonification system. Other

[1] Centro di Sonologia Computazionale, Department of Information Engineering, University of Padova

issues could not be solvable, for example a wheelchair presence management. However the core application of *SoundingARM*, that is the recognition of the gesture "point an object" is well working and the results obtained are more than encouraging.

Appendix A

Tables A.1, A.2, A.3, A.4, A.5 and A.6 are divided in 3 sub-tables: Clap, Opening and Audio. Clap contains information about the distance between the hands, the frame and the time when the clap occurs. Opening contains information about the distance, the frame when the hands are at the maximum distance each other, it contains also the velocity of the clap, calculated as ratio in the space of the interval of the distances and the time interval. Audio table list all the peaks found in the data, that is the noise of a clap. Lastly the delay is a simple subtraction between the value of the time in the Clap table and the Audio time value.

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	Frame	Time(s)	Dist.(mm)	Frame	Vel(m/s)	Sample	Time(s)	(s)
			1476.59	106				
35.57	136	4.53			1.44	193204	4.38	0.15
			1282.87	181				
35.57	206	6.87			1.50	298013	6.76	0.11
			1266.99	245				
79.42	279	9.30			1.05	399933	9.07	0.23
			1284.76	308				
55.85	346	11.53			0.97	497980	11.29	0.24
			1242.06	374				
84.44	411	13.70			0.94	594165	13.47	0.23
			1262.20	439				
79.92	475	15.83			0.99	693613	15.73	0.11
			1226.24	506				
39.60	539	17.97			1.08	786367	17.83	0.14
			1238.26	569				
65.60	602	20.07			1.07	879125	19.93	0.13
			1161.55	625				
105.44	658	21.93			0.96	963192	21.84	0.09
			1174.43	683				
70.37	723	24.10			0.83	1054408	23.91	0.19
			1190.48	747				
74.29	782	26.07			0.96	1146426	26.00	0.07
			1195.60	766				
66±22					1.1±0.2	**Average:**		0.15±0.06

Table A.1: Very Slow Applause.

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	Frame	Time(s)	Dist.(mm)	Frame	Vel(m/s)	Sample	Time(s)	(s)
			1451.20	61				
76.12	84	2.80			1.79	116221	2.64	0.16
			965.09	94				
73.57	103	3.43			2.97	147321	3.34	0.09
			971.87	117				
63.80	128	4.27			2.48	181400	4.11	0.15
			1006.57	140				
86.51	150	5.00			2.76	215765	4.89	0.11
			1022.90	163				
50.21	175	5.83			2.43	250335	5.68	0.16
			1042.42	187				
63.20	199	6.63			2.45	285248	6.47	0.17
			1017.84	211				
44.35	220	7.33			3.24	319861	7.25	0.08
			922.10	232				
111.84	241	8.03			2.70	351840	7.98	0.06
			954.89	256				
67.29	266	8.87			2.66	385949	8.75	0.11
			926.58	280				
56.36	291	9.70			2.37	421247	9.55	0.15

Table A.2: Normal Applause (Part 1).

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	Frame	Time(s)	Dist.(mm)	Frame	Vel(m/s)	Sample	Time(s)	(s)
			1451.20	61				
76.12	84	2.80			1.79	116221	2.64	0.16
			965.09	94				
73.57	103	3.43			2.97	147321	3.34	0.09
			971.87	117				
63.80	128	4.27			2.48	181400	4.11	0.15
			1006.57	140				
86.51	150	5.00			2.76	215765	4.89	0.11
			1022.90	163				
50.21	175	5.83			2.43	250335	5.68	0.16
			1042.42	187				
63.20	199	6.63			2.45	285248	6.47	0.17
			1017.84	211				
44.35	220	7.33			3.24	319861	7.25	0.08
			922.10	232				
111.84	241	8.03			2.70	351840	7.98	0.06
			954.89	256				
67.29	266	8.87			2.66	385949	8.75	0.11
			926.58	280				
56.36	291	9.70			2.37	421247	9.55	0.15

Table A.3: Normal Applause (Part 2).

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	**Frame**	**Time(s)**	**Dist.(mm)**	**Frame**	**Vel(m/s)**	**Sample**	**Time(s)**	**(s)**
			1007.71	303				
56.69	315	10.50			2.38	456182	10.34	0.16
			962.89	326				
63.98	337	11.23			2.45	489816	11.11	0.13
			989.67	350				
39.35	359	11.97			3.17	524606	11.90	0.07
			936.59	373				
61.06	383	12.77			2.63	558118	12.66	0.11
			827.56	395				
53.03	407	13.57			1.94	590540	13.39	0.18
			845.11	417				
22.93	431	14.37			1.76	623668	14.14	0.22
			483.21	444				
62.14	449	14.97			2.53	656046	14.88	0.09
			935.11	462				
85.44	472	15.73			2.55	689426	15.63	0.10
			949.28	485				
69.30	497	16.57			2.20	722507	16.38	0.18
			522.51	508				
64±19					2.5±0.4	**Average:**		0.13±0.04

Table A.4: Normal Applause (Part 3).

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	Frame	Time(s)	Dist.(mm)	Frame	Vel(m/s)	Sample	Time(s)	(s)
			1373.71	32				
58.32	48	1.60			2.47	64673	1.47	0.13
			951.01	57				
69.21	67	2.23			2.65	92135	2.09	0.14
			870.91	76				
25.57	84	2.80			3.17	118598	2.69	0.11
			918.65	94				
75.10	104	3.47			2.53	146252	3.32	0.15
			900.00	113				
66.49	121	4.03			3.13	173485	3.93	0.10
			795.20	131				
54.42	139	4.63			2.78	199163	4.52	0.12
			861.96	149				
41.00	158	5.27			2.74	225549	5.11	0.15
			870.61	166				
93.02	175	5.83			2.59	250966	5.69	0.14
			899.32	184				
77.28	192	6.40			3.08	277257	6.29	0.11
			981.19	202				
65.26	210	7.00			3.43	304752	6.91	0.09

Table A.5: Very Fast Applause (Part 1).

Clap			Opening			Audio Peaks		Delay
Dist.(mm)	**Frame**	**Time(s)**	**Dist.(mm)**	**Frame**	**Vel(m/s)**	**Sample**	**Time(s)**	**(s)**
			821.79	220				
62.71	229	7.63			2.53	330266	7.49	0.14
			762.97	238				
66.35	247	8.23			2.32	356493	8.08	0.15
			829.09	256				
59.73	265	8.83			2.56	383752	8.70	0.13
			729.74	275				
73.11	283	9.43			2.46	410619	9.31	0.12
			736.36	292				
45.10	300	10.00			2.59	437240	9.91	0.09
			776.89	312				
56.76	319	10.63			3.09	464125	10.52	0.11
			813.37	330				
60.34	338	11.27			2.82	492437	11.17	0.10
			759.65	348				
47.62	357	11.90			2.37	519881	11.79	0.11
			589.96	372				
61±15					2.7±0.3	**Average:**		0.12±0.02

Table A.6: Very Fast Applause (Part 2).

Appendix B

Evaluation of the down-sampling. Tables B.1 and B.2 represent the distance between original and sampled data, discussing in all the different possibilities for making this transformation. The five possibilities are the same offered by matlab.

The green cells contain the best values, while the red ones contain the worst ones. Counting every green cell as a score, the Mean Squared Error is minimized by *v5cubic* interpolation while the Signal to Noise ratio is maximized by linear interpolation; however the *v5cubic* interpolation collects more points and wins this challenge. We will use this kind of transformation for our further data analysis.

	Hips	Chest	Neck	Head	Head Site	Left Collar	Left Shoulder	Left Elbow	Left Wrist	Left Hand Site	Right Collar	Right Shoulder
nearest	5.1038	3.3003	0.2930	5.7460	3.4074	0.3093	5.7545	3.1119	0.3180	5.6272	3.2480	0.4490
linear	0.0265	0.0994	0.0132	0.0347	0.0935	0.0136	0.0620	0.1218	0.0144	0.1361	0.2313	0.0481
spline	0.0214	0.1805	0.0060	0.0263	0.1555	0.0062	0.0599	0.1887	0.0067	0.1427	0.3121	0.0436
cubic	0.0188	0.1200	0.0067	0.0235	0.1067	0.0068	0.0546	0.1402	0.0075	0.1323	0.2507	0.0411
v5cubic	0.0189	0.1335	0.0052	0.0235	0.1171	0.0053	0.0544	0.1496	0.0059	0.1318	0.2631	0.0400

Table B.1: Mean Squared error part 1. In rows there are the interpolation methods, in columns every joint of a skeleton. The values are the mean squared error introduced by the interpolation method in estimation of the position of the specified joint.

	Right Elbow	Right Wrist	Right Hand Site	Left Hip	Left Knee	Left Ankle	Left Foot Site	Right Hip	Right Knee	Right Ankle	Right Foot Site
nearest	5.2762	3.0288	0.5123	5.7545	3.1119	0.3180	6.6129	4.5147	0.6251	9.7673	8.1563
linear	0.0936	0.2194	0.0414	0.0620	0.1218	0.0144	0.0836	0.3304	0.1026	0.1240	0.2659
spline	0.0922	0.2897	0.0357	0.0599	0.1887	0.0067	0.0806	0.4235	0.1045	0.1156	0.3239
cubic	0.0843	0.2304	0.0340	0.0546	0.1402	0.0075	0.0745	0.3661	0.0984	0.1098	0.2804
v5cubic	0.0846	0.2451	0.0327	0.0544	0.1496	0.0059	0.0741	0.3734	0.0981	0.1063	0.2846

Table B.2: Mean Squared error part 2. In rows there are the interpolation methods, in columns every joint of a skeleton. The values are the mean squared error introduced by the interpolation method in estimation of the position of the specified joint.

Appendix C

OptiTrack MoCap Starting Procedure

Starting and calibration procedure for the *OptiTrack MoCap* system installed at IPEM laboratory of Gent University. This procedure is executed before starting an experiment.

1. Switch on master switch: red button in the multi-plug adapter, it is near the wall.
2. Turn on the computer of the *MoCap* system.
3. Loading windows and login.
4. Turn on the video projector using the remote control. The video projector is turned on when a green led is switched on. Otherwise, if the led one is orange, video projector is in stand-by, if all is turned off you have to find and connect the power cable. The video projector could not be needed for an experiment, in any case it is very useful during the calibration procedure and the skeleton detecting procedure. In fact, it is possible to follow all the procedure in the projection screen.
5. Computer Desktop: double click at icon 1.6 (Arena) to open the control program.
6. In this program: to begin with the calibration go to:

   Wizard →Calibration →Wand Type →1 Marker

   Calibration Type →Full Calibration

   There is a visualization of all the camera (12) this visualization is in a gray or green or red box for each camera, if respectively there are 0 or 1

or more markers in the camera view. (the other cameras could be mapped as markers, this is not a problem). All the camera must see the scene, otherwise it is needed to fix it, shifting or twisting the cameras, paying attention to do not touch the camera, but use the knobs instead. For watching the scene shoot by the camera, just right click on the box and select "Grayscale image": so now in the box there is the camera view (shifting this windows thought the border of the screen and it will appear projected in screen).

7. [Only if it happens] If, during the camera calibration, you've touched a camera, the system turns off all cameras, to protect itself. Reboot windows and return to point 3.

8. At the end of calibration, select "Block all visible points". These "blocked points" won't be used in processing of the markers.

9. Click on Next →Wand Data, choose: Slow (More Accurate).

10. Begin the calibration of the cameras, grasp the "Calibration Wand" and oscillate it form the right to the left, up and down onto the field. In the cameras boxes will appear a green trace, (otherwise click on the button), when the time is expired the trace became red.

11. Select: Data Point, Data: 300, Resolution: default and Min Camera: 11 and click on Start Calculation and wait for the calculation.

12. Locate the calibration square in the middle of your scene. This will be the (0,0,0).

13. On the Calibration Wizard →click Next →Select 3 markers of the calibration square, with the mouse →Click Set Ground Plane

14. Save the calibration

15. Save the project

16. To create the skeleton: Select Wizard / Skeleton →Next →choose Create: default with 34 markers →next.... (explanation)

17. Record a new T-Pose

18. Switch off time

19. Click on play, now it is possible observe the markers moving

20. Control and Fix the height and the shoulder →Click on Next →Auto Assign (relationship within the limbs) →Click on Next

21. Save the skeleton

22. [Optional] To make a reference surface (Define a Rigid Body): Wizard / Rigid Body →select with mouse all the markers →Click Save.

Bibliography

[1] M. Leman and R. I. Godøy. *Musical Gestures: Sound, Movement, and Meaning*. Routledge, London, 2009.

[2] J. Wapnick. A Review of Research on Attitude and Preference. *Bulletin of the Council for Research in Music Education*, 1(48):1–20.

[3] N. P. M. Todd. Motion in music: a neurobilogical perspective. *Music Perception*, 17(1):115–126, 1999.

[4] A. Friberg, J. Sunberg, and L. Frydén. Music from motion: sound level envelopes of tones expressing human locomotion. *Journal of New Music Research*, 29(3):199–210, 2000.

[5] A. Camurri, P. Coletta, M. Ricchetti, and G. Volpe. Expressiveness and physicality in interaction. *Journal of New Music Research*, 29(3):187–194, 2000.

[6] R. I. Godøy, E. Haga, and A. R. Jensenius. Playing "Air Instruments": Mimicry of Sound-producing Gestures by Novices and Experts. In S. Gibet, N. Courty, and J.-F. Kamp, editors, *Gesture in Human-Computer Interaction and Simulation, 6th International Gesture Workshop*, volume LNAI 3881, pages 256–267. Springer-Verlag, Berlin Heidelberg, 2006.

[7] M. Leman. *Embodied Music Cognition and Mediation Technology*. The Mit Press, Cambridge, 2007.

[8] Embodied Music Cognition. Web Site. Available at: `http://wikipedia.org` (Last check: 28 June 2012).

[9] M. Leman, M. Demey, M. Lesaffre, L. van Noorden, and D. Moelants. Concepts, Technology, and Assessment of the Social Music Game "Sync-in-Team". In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 04*, pages 837–842, Washington, DC, USA, 2009. IEEE Computer Society.

[10] A. Camurri, S. Canazza, C. Canepa, A. Rodà, G. Volpe, S. Zanolla, and G. L. Foresti. The Stanza Logo–Motoria: an Interactive Environment for Learning and Communication. In *Proc. of Sound and Music Computing Conference*, pages 353–360, Barcelona, July 2010.

[11] R. W. Massof. Auditory Assistive Devices for the Blind. In *Proc. of International Conference on Auditory Display*, Boston, June 2003.

[12] Atlas, 2003. Available at: `http://www.csun.edu/cod/conf/2003/proceedings/140.htm` (Last check: 28 June 2012).

[13] Talking Sings, 2000. Available at: `http://www.talkingsigns.com/tksinfo.shtml` (Last check: 28 June 2012).

[14] Y. Sonnenblick. An indoor navigation system for blind individuals. In *Proceedings of the 13th annual Conference on Technology and Persons with Disabilities*, 1998.

[15] L. Ran, S. Helal, and S. Moore. Drishti: An integrated indoor/outdoor blind navigation system and service. In *PerCom*, pages 23–32, 2004.

[16] S. Mau, N. A. Melchior, M. Makatchev, and A. Steinfeld. BlindAid: An Electronic Travel Aid for the Blind. Technical report, Pittsburgh, PA: Carnegie Mellon University, Robotics Institute, 2008.

[17] A. Hub, J. Diepstraten, and T. Ertl. Design and Development of an Indoor Navigation and Object Identification System for the Blind. *SIGACCESS Access. Comput.*, (77-78):147–152, 2004.

[18] C. Magnusson, M. Molina, K. Rassmus-Gröhn, and D. Szymczak. Pointing for non-visual orientation and navigation. *October*, pages 735–738, 2010.

[19] M. Zöllner, S. Huber, H. Jetter, and H. Reiterer. NAVI – A Proof-of-Concept of a Mobile Navigational Aid for Visually Impaired Based on the Microsoft Kinect. *Ifip International Federation For Information Processing*, 6949(c):584–587, 2011.

[20] S. Mann, J. Huang, R. Janzen, R. Lo, V. Rampersad, A. Chen, and T. Doha. Blind navigation with a wearable range camera and vibrotactile helmet. *Computer*, pages 1325–1328, 2011.

[21] B. Röder, W. Teder-Sälejärvi, A. Sterr, F. Rösler, S. A. Hillyard, and H J Neville. Improved auditory spatial tuning in blind humans. *Nature*, 400(6740):162–166, 1999.

[22] C. Leclerc, D. Saint-Amour, M. E. Lavoie, M. Lassonde, and F. Lepore. Brain functional reorganization in early blind humans revealed by auditory event-related potentials. *NeuroReport*, 11(3):545–50, 2000.

[23] R. Weeks, B. Horwitz, A. Aziz-Sultan, B. Tian, C. M. Wessinger, L G Cohen, M Hallett, and J P Rauschecker. A positron emission tomographic study of auditory localization in the congenitally blind. *Journal of Neuroscience*, 20(7):2664–2672, 2000.

[24] A. G. De Volder, H. Toyama, Y. Kimura, M. Kiyosawa, H. Nakano, A. Vanlierde, M. C. Wanet-Defalque, M. Mishina, K. Oda, K. Ishiwata, and Et al. Auditory triggered mental imagery of shape involves visual association areas in early blind humans. *NeuroImage*, 14(1 Pt 1):129–139, 2001.

[25] N. Lessard, M. Paré, F. Lepore, and M. Lassonde. Early-blind human subjects localize sound sources better than sighted subjects. *Nature*, 395(6699):278–280, 1998.

[26] C. Muchnik, M. Efrati, E. Nemeth, M. Malin, and M. Hildesheimer. Central auditory skills in blind and sighted subjects. *Scandinavian Audiology*, 20(1):19–23, 1991.

[27] P. Ghesquière, J. Laurijssen, W. Ruijssenaars, and P. Onghena. The significance of auditory study to university students who are blind. *Journal of Visual Impairment Blindness*, 93:41–45, 1999.

[28] M. Noisternig and C. Frauenberger. 3D Audio Interfaces for the Blind. *Proceedings of the 2003 International Conference on Auditory Displays*, pages 1–4, 2003.

[29] M. Kitagawa and B. Windsor. *MoCap for Artists: Workflow and Techniques for Motion Capture*. Focal Press, 2008.

[30] R. Donadi. Music Paint Machine: an Interactive Music System for Educational and Artistic Purpose. Master's thesis, University of Padova, 2011.

[31] W. Zeng and Z. Zhang. Microsoft Kinect Sensor and Its Effect. *Ieee Multimedia*, 19(2):4–10, 2012.

[32] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth Mapping Using Projected Patterns, 2010.

[33] Z. Zalevsky, A. Shpunt, A. Maizels, and J. Garcia. Method and System for Object Reconstruction, 2007.

[34] J. Kramer, C. Parker, D. Herrera, N. Burrus, and F. Echtler. *Hacking The Kinect*. Apress, 2012.

[35] NITE Controls User Guide. Web Site. Available at: `http://www.openni.org/` (Last check: 28 June 2012).

[36] Mental Chronometry. Web Site. Available at: `http://en.wikipedia.org/wiki/Mental_chronometry#Types` (Last check: 28 June 2012).

[37] R. J. Kosinski. *A Literature Review on Reaction Time*. Clemson University. Available at: `http://biae.clemson.edu/bpc/bp/Lab/110/reaction.htm` (Last check: 28 June 2012).

[38] Kinect. Web Site. Available at: `http://en.wikipedia.org/wiki/Kinect` (Last check: 28 June 2012).

[39] C. Thinus-Blanc and F. Gaunet. Representation of space in blind persons: vision as a spatial sense? *Psychological Bulletin*, 121(1):20–42, 1997.

[40] T. J. Carroll. *Blindness: What it is, what it does, and how to live with it.* Little, Brown, Boston, MA, 1961.

[41] A. Wagner-Lampl and G. W. Oliver. Folklore of blindness. *Journal of Visual Impairment and Blindness*, 88:267–276, 1994.

[42] D. H. Ashmead, E. W. Hill, and C. R. Talor. Obstacle perception by congenitally blind children. *Perception And Psychophysics*, 46(5):425–433, 1989.

[43] J. J. Rieser. Blindness and brain plasticity in navigation and object perception. *New York, NY: Lawrence Erlbaum Associates/Taylor Francis Group.*, 2008.

[44] M. Ohuchi, Y. Iwaya, and T. Munekata. Cognitive-Map Formation of Blind Persons in A Virtual Sound Environment. *Special Education*, pages 1–7, 2006.

[45] T. H. Riehle, P. Lichter, and N. A. Giudice. An indoor navigation system to support the visually impaired. *Conference Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, pages 4435–4438, 2008.

[46] L. B. Merabet and J. Sánchez. Audio-Based Navigation Using Virtual Environments: Combining Technology and Neuroscience. *AER Journal Research and Practice in Visual Impairment and Blindness*, 2(3):1–12, 2001.

[47] M. C. Wanet and C. Veraart. Processing of auditory information by the blind in spatial localization tasks. *Perception and Psychophysics*, (38):91–96, 1985.

[48] S. Zanolla, F. Romano, F. Scattolin, A. Rodà, S. Canazza, and G. L. Foresti. When sound teaches. In S. Zanolla, F. Avanzini, S. Canazza, and A. de Götzen, editors, *Proceedings of the SMC 2011 - 8th Sound and Music Computing Conference*, pages 64–69, 2011.

[49] OpenNI Hardware. Web Site. Available at: `http://www.openni.org/hardware`.

[50] Microsoft's move, Tech giant spends big to launch Wii rival Kinect. Web Site. Available at: `http://www.nypost.com/p/news/business/microsoft_move_3gVmAyryJuD6px1dV7LeDP` (Last check: 28 June 2012).

[51] Kinect Confirmed As Fastest-Selling Consumer Electronics Device. Web Site. Available at: `http://community.guinnessworldrecords.com` (Last check: 28 June 2012).

[52] Kinect Sales Surpass Ten Million. Web Site. Available at: `http://www.xbox.com/en-us/press/archive/2011/0308-ten-million-kinects` (Last check: 28 June 2012).

[53] Kinect for Windows SDK. Web Site. Available at: `http://www.microsoft.com/en-us/kinectforwindows/` (Last check: 28 June 2012).