UAlg ISE
UNIVERSIDADE DO ALGARVE
INSTITUTO SUPERIOR DE ENGENHARIA

UNIVERSITY OF THE ALGARVE

# *INTELLIGENT SYSTEM FOR INTERACTION WITH VIRTUAL CHARACTERS BASED ON VOLUMETRIC SENSORS*

RICARDO MARTINS ALVES

Thesis

**Master Thesis in Electric and Electronic Engineering**

Work done under the supervision of:
Prof. Doutor João Miguel Fernandes Rodrigues and Prof. Roberto Lam

2015

# *INTELLIGENT SYSTEM FOR INTERACTION WITH VIRTUAL CHARACTERS BASED ON VOLUMETRIC SENSORS*

Declaration

I hereby declare to be the author of this work, which is original and unpublished. Authors and works consulted are properly cited in the text and appear in the included reference list.

*Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.*

Ricardo Alves

# Abstract

One of the most interesting challenges nowadays is the interaction between human and machine (Human-Computer Interaction), namely, how to create a natural language between machine and humans. This can be achieved through the analysis of the environment, human poses and gestures using computer vision techniques. The appearance of three-dimensional (3D) depth sensors in the consumers market at a very competitive prices, brings us one step closer, giving us another tool to achieve the above goal. Those sensors can be used in different fields such as video/film interaction, interfaces, gaming and virtual and augmented reality.

This thesis focus on the development of (intelligent) systems for interaction with virtual characters or films based on volumetric sensors (in this case Microsoft Kinect). Several contributions are presented: (a) for videos/film interaction, users can trigger different narratives interacting through a 3D sensor, detecting different movements and gestures that change the film with a set of previously filmed scenes, all according to a previous written storyboard. The system is presented using rear-projection (real sized human projection) and an holographic technique (smaller size).

Some of those techniques can also be used for an (b) interactive interface, with two different developed installations featuring a virtual host working as a public relations (PR). Both solutions are capable of interacting with users through intuitive gestures, extracting users' face and body, useful for promotional "gift", and tracking over time

their joints and position, building key statistics about interacted content, users' actions and favourite menus. While one installation is limited to the (b.1) captured angle of one sensor, 57º (degrees) and for visualization uses holographic representation, the other (b.2) offers the possibility to cover all the area in from of the installation, 180º while for visualization using rear-projection.

Also focusing an interactive interface, (c) a 360º installation was developed, which analyses the space surrounding it with the same characteristics of b), but can also validate if the user(s) is(are) the same if he/she/they left the installation for some time and came back again. The system can also know where each user is in the 360º field of view in front of the installation (the user can move freely all around the installation), creating and updating the statistics of all detected users.

Related to gaming, is presented (d) an augmented reality tool for pool, snooker or billiards. A 3D sensor, placed above the table alongside a projector, is used to detect game elements on-the-fly: balls, cue and borders of the table. Combining the extracted information, trajectories are predicted with the result being visible onto the table through the projection, helping especially beginners and amateur players.

**Keywords:** Interactive Installation, Computer Vision, Human-Computer Interaction, Applications, Gesture Recognition, Holography, Visualization, Applications, Kinect, Natural Interaction, Pool Game, Augmented Reality.

# Resumo

A tecnologia vem sendo desenvolvida para ajudar-nos a completar ou aumentar a produtividade nas nossas tarefas diárias. Muitas das máquinas construídas têm sido progressivamente aperfeiçoadas para funcionar mais como um ser humano, usando para isso os mais variados sensores. Um dos problemas mais desafiantes que a tecnologia encontrou é como dar a uma máquina a capacidade que um "animal" tem de perceber o mundo através do seu sistema visual. Uma solução será usar na máquina sistemas inteligentes que usem visão computacional. Uma grande ajuda pode chegar da perceção de profundidade pela máquina, tornando menos complexa a deteção e a compreensão de objetos numa imagem por parte desta. Com o aparecimento de sensores volumétricos (tridimensional 3D) no mercado consumidor, aumentaram os desenvolvimentos feitos nesta área científica, permitindo assim a sua integração na maioria dos dispositivos, tais como computadores ou dispositivos móveis, a um preço muito competitivo. Os sensores volumétricos podem ser usados nas mais variadas áreas pois apesar de terem aparecido inicialmente na área dos videojogos, estendem-se ainda à área de vídeo, modelação 3D, interfaces, jogos ou realidade virtual e aumentada.

Esta dissertação foca essencialmente no desenvolvimento de sistemas (inteligentes) baseados em sensores volumétricos (neste caso a Microsoft Kinect) para a interação com avatares ou filmes. Quanto a aplicações na área de vídeo, foi desenvolvida uma

solução onde um sensor 3D ajuda um utilizador a seguir uma narrativa que é iniciada assim que o utilizador é detetado, mudando os acontecimentos do vídeo consoante ações pré-determinadas do utilizador. O utilizador pode então mudar o rumo da história mudando de posição ou efetuando um gesto. Esta solução é ilustrada utilizando retroprojeção, existindo ainda a possibilidade de ser apresentada em modo holograma numa abordagem à escala.

O descrito no anterior parágrafo pode também ser aplicada a uma solução de vertente mais comercial. Para isso, foi desenvolvido uma aplicação altamente configurável, podendo-se ajustar (em termos visuais) às necessidades de diferentes companhias. O ambiente gráfico é acompanhado por um avatar ou por um video (previamente gravado), que interage com um utilizador através de gestos, dando uma sensação mais realista devido à utilização de holografia. Ao interagir com a instalação, são registados todos os movimentos e interações efetuadas pelo utilizador para que estatísticas sejam construídas, de maneira a perceber os conteúdos com mais interesse bem como as áreas físicas com mais interação. Adicionalmente, o utilizador poderá ter a sua fotografia completa ou tipo BI extraída, podendo-lhe ser oferecidos em produtos promocionais da empresa. Devido à curta área de interação oferecida por um sensor deste tipo (Kinect), foi também desenvolvida a possibilidade de juntar vários sensores, 4 para cobrir 180º (graus) em frente da instalação ou ainda 8 para cobrir os 360º à volta da instalação, de maneira a que os utilizadores possam ser detetados por qualquer um deles e que não sejam perdidos quando atravessam para uma zona de outro sensor, ou mesmo quando saem do campo de visão dos sensores e retornam mais tarde.

Apesar dos sensores referidos serem mais conhecidos na interação com um jogo virtual, jogos reais e físicos também podem beneficiar deste tipo de sensor. Neste último ponto, é apresentada uma ferramenta de realidade aumentada para *snooker* ou bilhar. Nesta aplicação, um sensor 3D colocado por cima da mesa, capta a área de jogo sendo depois processada para que sejam detetadas as bolas, o taco e as tabelas. Sempre que possível, esta deteção é feita usando a terceira dimensão (profundidade) ofer-

ecida por estes sensores, tornando-se por exemplo mais robusto a mudanças quanto a condições luminosas. Com estes dados é então previsto, utilizando álgebra vetorial, a trajetória da bola, sendo projetado o resultado na mesa.

**Palavras-chave:** Instalação Interativa, Visão Computacional, Interação Homem-Máquina, Reconhecimento de Gestos, Holografia, Visulização, Aplicações, Kinect, Interação Natural, Snooker, Realidade Aumentada.

# Acknowledgments

Present in this thesis is the work of many people who spent their time to spread their knowledge onto me. Many of this people may be left aside from my memory, but their knowledge will be carried alongside my big thanks for all they've done. A big thanks for all my recent colleagues, from my project co-workers, Luís Sousa and Aldric Negrier, who more than anyone helped me working on it, to my workplace colleagues, Daniel Martins and António Belguinha, who shared with me part of their knowledge on their free time by working with me on other projects. My friends and family also deserve a special thanks. Although they gave no direct contribution to the work present here, they allowed me to focus on it by cooking and caring for me as my mother did, but also by spending part of their lives enjoying it with me and helping me to clear my mind. Finally, a special and very big thanks to Professor João Rodrigues and his orientations. Without him, the present work would not be finished any time soon.

# Table of Contents

# 1

# Introduction

Over the last few decades technology improvements allowed the human race to create several devices that changed our lives completely, helping us in all types of tasks. For example a simple alarm clock help us starting the day, while our meals are prepared in a technology advanced kitchen. Our car help us reach our workplace, where internet, computers and other advanced equipment helps turning our tasks easier and more productive.

Machines and computers are not only being designed to help us in all kind of tasks, but also to see the world the way humans do, turning them into machines capable of adapting themselves to several types of works and interacting with us the way we do with our colleagues and friends. Computer Vision (CV) is being widely used to

cover all tasks where a computer needs to capture the world as our eyes would, but computing a 2D world, obtained through a monocular camera, is not everything our eyes see. Although there are some tasks where a 2D image is enough, more complex tasks need more complex algorithms, whereas complex algorithms need more time to be completed and, in cases where real time processing is needed, time is an important resource. We can always use stereo cameras, but nowadays, despite outstanding improvements in the outputs (depth) returned from these cameras, they are not yet as good as modern three-dimensional (3D) depth sensors, such as e.g., Microsoft Kinect (Kinect, 2014b), Leap Motion (Motion, 2014) or the Structure Sensor (Structure, 2014).

Over the last few years with the popularization of 3D depth sensors and the decreasing of their prices, one more tool is available to helps us in the development of a computer that can "see" the 3D world the same way humans do. Of course that processing the "3D world" has a lots of challenges but new possibilities emerges.

## 1.1 Scope of the Thesis

Since 3D Depth sensors appeared in the consumers market, academic community started researching and developing algorithms to take advantage of the fact that computers could now, at an accessible price, use (more) the third dimension of our world. The current thesis focus on research and development (R&D) using these sensors, i.e., mainly in the R&D of (intelligent) systems for interaction with virtual characters or films based on 3D depth sensors (in this case Microsoft Kinect).

The thesis is integrated in the financed QREN I&DT project: "PRHOLO: The holographic realistic public relations" (QREN I&DT, number 33845), with promoter SPIC - Creative solutions and co-promoter University of the Algarve. The goal is to develop a human sized structure with a virtual or filmed holographic character (in its interior), giving depth perception to users. This structure is equipped with 3D depth sensors endowing the whole installation with natural interaction (NI) capabilities with its users,

allowing them to navigate between menus that are easily adaptable and extensible to any event or company that owes it. While interacting with users, according to the company needs, the installation can collect and track several information about users as well as requested content for statistical purposes, accessible afterwards through a database only visible to the company.

Also integrated in the focus of the thesis, intelligent systems for interaction, is PoolLiveAid, a project (not financed) of the author of this thesis in conjunction with his colleague Luís Sousa, being developed since 2012 and started on the first master's degrees year on Computer Vision (CV) subject. PoolLiveAid goal consists on the prediction of snooker or billiards moves in real time, to help experienced uses. The first version worked with a Full HD web-cam and a projector, both placed above the table. Many limitations were found, mostly due to variation of lightning conditions. With the knowledge obtained from PRHOLO project, a more stable and reliable version was built using a 3D depth sensor instead of the webcam, initially two Microsoft Kinect sensors, and now a Microsoft Kinect 2.

## 1.2 Objectives

The objective of this thesis is to develop systems based on the Microsoft Kinect (3D depth) sensor, with the initial goal consisting in the development of: (i) a commercial interactive installation (PRHOLO) and (ii) a augmented reality real time snooker tool (PoolLiveAid). Also developed was (iii) an art installation. All of the previous applications must work in real time and take advantage of the third-dimension given by this sensor, with specific goals being described bellow:

**Commercial interactive installation:** Taking advantage of a holographic technique, this installation has an avatar working as host, which can respond to users sound input and gestures. With the help of a highly configurable menus, the user can navigate in the customizable options, while the installation tracks received inputs to create statis-

tics. The main goals for this application are:

- Creation of a customizable interface;
- Detection and tracking of users;
- Reaction from the installation through sound input;
- Recognition of users gestures;
- Creation of statistics through collected data;
- Allowing the use of multiple sensors to cover a wider area.
- Recognition of previous users.

**Real time snooker tool:** An augmented reality tool to help inexperienced or beginner players of snooker, pool or billiards, by detecting the key elements of the game, predicts the trajectory of the ball showing the result in real time onto the table. The main goals are:

- Detection of table borders;
- Detection of playable balls;
- Distinguishing the white ball;
- Detection of the cue;
- Shot detection;
- Computing if the white ball is going to be hit;
- Trajectory calculation.

**Art installation:** Consists on a real sized rear projection or a small holographic installation capable of interacting with users using Natural Iteration (NI) by changing the narrative. The main goals are:

- Detection and tracking users position;
- Recognition of defined gestures;
- Triggering changes on the storyboard.

## 1.3 Structure of the Thesis

The structure of the writing of this thesis consists on the compilation of the papers where the above subjects were deepened, nevertheless more papers were written during the master and are listed in Section 7.2. Each paper/chapter is presented as published or submitted, being discussed individually in a complete dedicated chapter, having their own introduction, state of art, methods and conclusions. The bibliography was removed from each individual paper, to simplify the reading and is presented in the end of the thesis instead.

Completing the paragraphs above, and to be completely clear of the work done by the author of this thesis, is described bellow an overall contribution to each of the 5 papers that corresponds to the 5 main chapters of this thesis:

- **Alves, R.**, Madeira, M., Ferrer, J., Costa, S., Lopes, D., Mendes da Silva, B., Sousa, L., Martins, J., Rodrigues, J.M.F. (2014) ***Fátima revisited: An interactive installation***, In Proc. Int. Multidisciplinary Scientific Conf. on Social Sciences and Arts, Varna, Bulgaria, 2-9 Set., pp. 141-148

  Contributions to this paper consists on the use of a 3D Depth Sensor (Kinect (2014b)) to track users onto a list, tracking/recording all skeletal and positional information over time, triggering different reactions of the hologram by analysing the stored information as well as extracting the user's face (see mainly Section 2.6).

- **Alves, R.**, Sousa, L., Negrier, A., Rodrigues, J.M.F., Cardoso, P.J.S., Monteiro, J., Gomes, M., Bica, P. (2015) ***PRHOLO: Interactive Holographic Public Relations***,

In Proc. 3rd Int. Conf. on Advances in Computing, Communication and Information Technology, Birmingham, UK, 26-27 April., pp. 124-128

Contributions to this paper continues the work done on the above paper: users are tracked onto two lists, one for the current active users and others for the previously tracked but now lost users. Using the stored information of users position, heat maps are built for a user or a group of users for statistical purposes, while also tracking users' head direction relative to the camera to estimate a looking direction of a user and counted the time spent looking to that specific direction. As well as extraction of users' head, is also performed a full body extraction of a user. Relative to the two lists implemented, a very simple algorithm recovers information of a lost user if the sensor is suddenly blocked (see mainly Section 3.3.1).

- **Alves, R.**, Negrier, A., Sousa, L., Rodrigues, J.M.F., Felisberto, P., Gomes, M., Bica, P. (2015) ***Interactive 180º Rear Projection Public Relations***, Procedia, vol. 51, pp. 592–601

  Contributions to this paper also continues the work done on the above paper: instead of one 3D depth sensor, is now managed incoming data of four different sensors, positioning in such way that the total angle covered is $180^o$ (see Section 4.3). Data is now stored into two lists for each camera, working similar to the explained above, while two main lists are built for all four cameras (see Section 4.3.1. Using the physical position of each sensor, an algorithm was implemented to distinguish each detected user by different sensors and enabling navigation between sensors without being lost (see Section 4.3.2).

- **Alves, R.**, Sousa, L., Negrier, A., Monteiro, J., Cardoso, P.J.S., Felisberto, P., Bica, Rodrigues, J.M.F. (2015) ***Interactive 360º Holographic Installation***, submitted to The Visual Computer.

  Contributions to this paper continues the work done one last time. Users can

now be seen in an area of 360º using eight 3D depth sensors with the improvement of recovering users when they return to the installation some time later (see Section 5.5.1 and Section 5.5.1), by using skeleton segments descriptors defined by length and colour of these segments. Some small changes were also done to other algorithms such as heat maps.

- Sousa, L., **Alves, R.**, Rodrigues, J.M.F. (2015) ***Augmented reality system to assist inexperienced pool players***, submitted to The Visual Computer.

  Contributions to this paper consists on a equal part of Section 6.5.2, detailing with other colleagues the global functioning of the system such as flowchart, layout and hardware, using a 3D depth sensor and a projector. With this discussed, work was done in detection of motion (see Section 6.5.3), triggering different phases of the game and other contributed algorithms, such as cue and ball detection (see Section 6.5.4 and Section 6.5.5).

## 1.4 Overview of the thesis

In the present chapter the theme was introduced as well as the main goals, contributions and the scope of this thesis, consisting on five different submitted or published papers related to 3D depth sensors. Chapter 2 presents an art installation with a storyboard that changes according to users position and movements, based on the inputs a Kinect returns from users body. Chapter 3 is an extension of the work done in Chapter 2, implementing a more robust system on a customizable interface with commercial purpose, reading and tracking users movement and position building useful statistics. Similar to this, Chapter 4 continues the work done with the main difference being the use of four 3D depth sensors working together to capture a wider area, increasing the interacting area of users by tracking them and recognizing them even when they cross between two Kinect areas. Finally in Chapter 5 an improvement was done in relation to the previous chapter, enabling an interaction area of 360º using 8 Kinect

sensors, with the addition of users recognition when later returning to the installation. In Chapter 6 is introduced a framework that works in real time, processing a game of *snooker* or billiards, detecting balls, borders and the cue to predict a trajectory, showing the result by projecting it onto the table. Chapter 7 is the final chapter, concluding this thesis and all work done in the previous chapters. As stated, some of the Chapters continue the work done up to that point and for that reason, some Sections are very similar with Section 3.3.1 Spatial Information, Gestures Recognition and Face and Body Extraction blocks being very similar with Section 4.3.1 and the same blocks, with the same happening with Section 5.5.1 and these very same blocks.

# 2

# Fátima Revisited: An Interactive Installation

## 2.1 Abstract

Three very young shepherds, on May 13, 1917, reported seeing "*... a lady even brighter than the sun...*", floating a meter or so in the air, near an old oak tree, when they were pasturing their little herd in *Cova da Iria*, Portugal. The story of *Our Lady of Fátima* has remained one of the most remarkable odes in Portuguese folklore. It is, beyond the religious event in itself, a key episode in the official history and culture of the Portuguese people. It is a day celebrated every year; more among the faithful believers,

but also in the media and even as a political catchphrase, widespread, recognized and celebrated, including by the Vatican and several Popes. In this Chapter, we present two multimedia installations where the central figure is *Our Lady of Fátima*, following two main ideas: (a) a "door" opens up the possibility to access multiple space-time experiments, and (b) we try to reach into the Portuguese imaginary surrounding the appearance of Our Lady of Fátima, using a female archetype suggesting a "sacred apparition". Both installations use a Microsoft Kinect sensor to detect the presence of a viewer for triggering the start of the narrative, followed by recognizing different movements of the viewer(s), e.g., moving left, right, forward, backward, etc., to create different flows in the presented narrative – even at some point changing the face of *Our Lady of Fátima* with the face of the viewer. Both installations use similar hardware and software concepts, except that the first one uses a Rear Projection solution where the narrative is presented by the image of Our Lady in real size and the second uses a Hologram.

## 2.2 Introduction

On May 13, 1917, three children, *Lucia de Jesus dos Santos* (10 years old), *Francisco Marto* (9 years old) and *Jacinta Marto* (7 years old), reported seeing "*...a lady even brighter than the sun...*" on an oak tree a meter or so high, when pasturing their little herd in the *Cova da Iria*, Portugal. This was the initial story of Our Lady of Fátima, following her appearance in this hamlet at the centre of the typical rural Portugal, in the early years of the 20th century. This was and still is, one of the most remarkable odes in folklore and sacred literature, with a large sanctuary built in her behalf (SantuárioFátima, 2014).

In the literature there are several interactive installations that integrate art, technology, image, film and volumetric sensors (Lee et al., 2014; Godbehere et al., 2012; Doyle and Jun, 2013), with also analytic frameworks to evaluate such interactions in

public installations (see e.g. Mathew et al. (2011)). In this Chapter we propose two installations, where the central figure is a "holy spirit" like figure that tries to recreate the sacred apparition of Our Lady of Fátima. Both installations use a Microsoft Kinect sensor to detect the presence of a person to trigger a narration. Different actions of the spectators (e.g., moving left, right, forward, backward, rising the arm, etc.), create different interactions in the narrative, including changing the face of the interlocutor with the face of the figure. Both installations are similar except, one uses a Rear Projection solution where the narrative is presented by a figure in real size (1.70 m height) and the second uses a Hologram (0.30 m height).

In the following Section it will be presented briefly the Our Lady of Fátima history, followed by a brief explanation of the Rear Projection and Hologram techniques. The next Section consists details both the hardware and software used, including how the films were prepared and the interaction with the Kinect was implemented. The final Section concludes with a small discussion.

## 2.3 Our Lady of Fátima History and Projection Techniques

On May 13, 1917 a little before noon, three children were prayed the Rosary, as it was their custom, as it was said, they were building a small stone house (seemingly on the site where today stands Basilica de Fátima (SantuárioFátima, 2014)). All on the routine of a bucolic setting: 3 children, a small herd, the backdrop of a rural country, sparsely populated and undeveloped.

The most consensual reports speak of the emergence of a sudden bright light, announced by a thunder. The little shepherds decided to seek shelter; however, soon after, a new pulsing bright light emerged, and this time even more intense. The surrounding area was inundated by a bright glow and by a distinguished apparition on top of a small holm oak (which is currently a sacred place, near where the Chapel of

the Apparitions is (SantuárioFátima, 2014)): "*...a lady even brighter than the sun, from whose hands hung a white rosary...*".

The Lady told the 3 children that they had to pray a lot and offer unreserved devotion to faith. The Lady then invited them to return to *Cova da Iria* during five consecutive months, always on the 13th of each month and always at the same hour – when the sun reaches the highest point in the sky. And so it was, the children returned, and the apparition of Fátima returned to *Cova da Iria* on the 13th of June, July, September and October (the exception was August; the apparition took place in a place called *Valinhos*, this probably because, on this day, the children had been taken to *Vila Nova de Ourém*.

Of course, since its first appearance, the story told by the little shepherds spread widely. According to historical reports, in her last appearance, on 13th October, there were present about 70.000 persons to receive the Lady, and in this last appearance the Lady said she was the *Lady of the Rosary* and that there, at the *Cova de Iria*, should be erected a chapel in her honour. For more details and illustrations about the story see (AssociaçãoDevotos, 2014; ImmaculateHeart, 2014).

Knowing the story of Our Lady, two projection techniques presented suitable for the installation: (a) *Rear Projection* (back-projection) is a technique well known and widely used (ProDisplay, 2014), as it consists in a projector positioned behind a screen casting a reversed image of the background or scene. This requires a large space to project (nevertheless, this can be minimized by using ultra-short-distance projectors), as the projector had to be placed at a minimum distance from the back of the screen, with the screen requiring a special retention surface (ProDisplay, 2014). The great benefit of this technique is that all the projection (and interaction) equipment can be completely hidden behind the screen, greatly increasing the belief that the user is interacting with a "sacred apparition". (b) *Holography* is a technique for recording interference patterns of light which can generate or display images in 3D (three dimensions) (Mihaylova, 2013). Sometimes, a hologram is also defined as a photographic image that

appears to have depth. In this case, holograms work by creating an image composed of two superimposed 2D pictures of the same object seen by different reference points. The use of slightly offset reference points is designed to mimic the image interpreted by the human brain, which likewise receives a distinct, slightly offset image from each eye that the brain combines into a coherent 3D image (Moser, 2014). One of the most common techniques to generate holograms is called the *Pepper's Ghost* (Sprott, 2006). In its basics, it uses a large piece of glass at a $45^o$ degrees angle to the audience and special lighting techniques for showing the audience a combination of light passing through from behind the glass and light reflecting off the glass, at a $90^o$ angle from the line of sight, creating the illusion that the reflected light appears from nowhere. The holographic effect is actually an object or image hidden from the audience and reflected off the screen. Spotlights with dimmer controls can alternately illuminate the area behind the glass or the area off the side. One light is turned up while the other was turned down, in such a way that the total light intensity was nearly constant. The best effect was achieved by using dark backgrounds. An example applied to the theatre and holography illustrating the entire length of the technique can be found in Rennie (2014); for the mechanical engineering parts see Figueiredo et al. (2014a). In the prototype used for this Chapter we used a very thin acrylic sheet (for a more realistic result a Mylar polyester film can be used) in a $45^o$ angle over a monitor (27"). The best benefit of this technique is that the spectator can interact with the installation as a 3D "sacred apparition" that is floating in the air, while also viewing all that is happening in the background and around the figure.

## 2.4 The Installation

The installation has two fundamental key ideas behind: one, where a door would be a possibility to access multiple space-time experiments, and the other, condensing the Portuguese imaginary as the female archetype as a sacred apparition, known as the

appearance of Our Lady of Fátima. The second idea was imposed immediately as more impactful, which reinforces the belief that, even given the provision of advanced technological Instruments, archaism survives and overlaps as the dominant reference.

Having Our Lady of Fátima history as background, we propose two installations: The first, with the dimensions of a real person with around 1.70 m. In this installation the story is rear-projected into an acrylic with a film retention, suspended around 0.5 m from the floor, representing a "time-space" door, and uses a projector of around 3000 lumens. In the second installation we use a hologram, currently still a prototype installation that allows holograms around a maximum of 30 cm height. Both installations share the same storyboard, differentiating only on the projection technique. The spectators, like the shepherds, are the visionaries of today. Generically, the story fades from a black background with a sputter until the holy figure appears. The sputtering sounds were mentioned in all the texts at that time, e.g., "*...tinnitus, as of horseflies in a jar...*" or "*...the crackling...*". So, it is also introduced the mist and thunder said to have been heard in the apparition. In parallel, the sound of a heart beating seemed ideal to delineate the appearance of the Virgin, as well as a church choir to reinforce the "divine ambience". The entire storyboard in general shares several key elements like the movements of the figure's hands and arms, head and eyes, seemingly launching the unexpected, in an aura of mystic awe; also a message is spoken from front to back, both fussing and frightening. A music from Robert Ashley, "Automatic writing" also goes along with the story.

All films (images) were purposely white saturated and flicker from time to time (see Fig. 2.1, and the Film Production Section) – this was to convey a bit the impact that the shepherds had. In Fig. 2.1 (left), it is shown a frame from the film used in the Rear Projection installation, and on the right, two frames tested on the holographic installation. As soon as any spectator passes by, the story and interaction starts, i.e., the presence of a spectator, triggers the storyboard (see Fig. 2.2, top block). The installation is then ready to interact with the spectator near it. In summary, there are six main

interactions (see Fig. 2.2). If the spectator:

- (a) moves/walks from left to right in front of the installation: the holy figure looks very fast to the right executing some movements and sounds; after the interaction, it returns to stage (e);
- (b) moves/walks from right to left in front of the installation: similar to a), but now it looks to the left, etc.;
- (c) moves the arms (e.g. rising them in an prayer gesture): The figure makes several movements, e.g. looks to sky, to the left and right, open-close arms, etc., accompanied with different sounds. This interaction finishes to stage (e);
- (d) moves closer to the installation: the face of the spectator is automatically segmented (cropped from each frame acquired in real time from the user) and overlapped with the one from the figure. When the spectator moves away, this overlapping finishes, returning to stage (e);
- (e) was still for some time: in this case there are 3 interactions that occur in sequence, depending on how much time the spectator remains still. (i) The figure was still, only sound changes. After a while if the spectator keeps still, (ii) the figure makes some movements like it was "speaking" with the spectator and (iii), the same as in ii), but different movements and sounds.
- (f) moves away: the figure also moves away, and all the images and sounds from the installation disappear, until a new spectator was presented.

## 2.5 Film Production

The film production was made in a Chroma-key studio to ensure the absence of spatial references and facilitate the post-production. The main concern was the angle of view since the subject (Fátima) should occupy all the field of view. Plus, the shot had to

be vertical to fit a human size projection. In these sense, it was used a full shot to insure the complete view of the subject. The illumination was made with three light sources: two key lights (to attenuate the relationship between shadows and light) and one backlight (to facilitate trimming). It was also used a smoke machine that created artificial fog and helped create a mystic atmosphere.

Concerning the post-production, it was developed into two distinct aesthetic experiences: The first one involves the subject in a heavenly aura characterized by overexposure while the second one clearly cuts the subject giving a greater degree of image definition and a greater sense of closeness. It was also added a drop-out effect, characterized by a small loss of data in the image. This effect refers to something that happened in the past (and simultaneously in the present) from the age of the older tapes, dating to the analog era. However the effect is not just aesthetic – it creates a jump cut that will facilitate the assembly of the different parts of interactivity.

The sound was thought to create additional stimuli to image. Sound effects and sound track have the ability to create audible conjunctures and appeal to the spectator's immersion into the narrative, against a compliant and passive attitude. According with Deleuze (1990), images and sounds no longer need to be based on movement, nor in a temporal linear sequence of past, present and future. So the sensory-motor sensations, time indirect representations, tend to be replaced by exclusively visual and audible conjunctures, namely the *opsign* and *sonsign*, time direct representations.

## 2.6 Kinect Interaction

Kinect (Kinect, 2014b) provide us three main streams to our interaction stages: RGB, which was in our case used to extracted a spectator face, depth, which was also used to extract a spectator face and was implicit called by Kinect to track users, and skeleton tracking, which was where we rely to detect if and how many spectators were interacting.

Figure 2.1: Left, one frame used in the film for the Rear Projection installation, and right two different frames used in the films on the holographic installation.

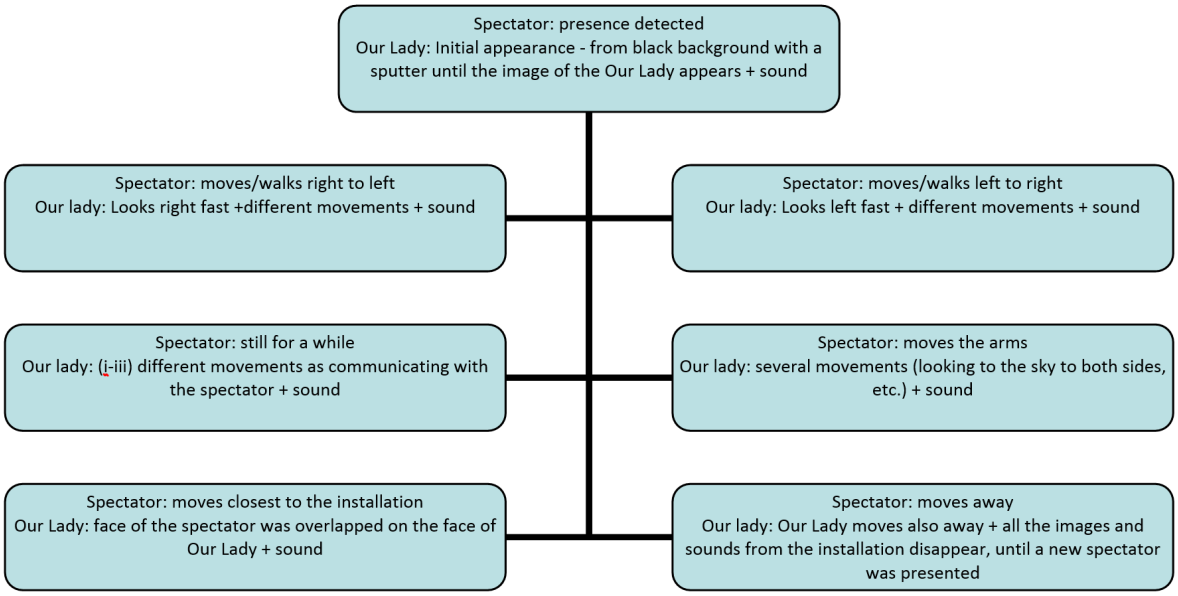


Figure 2.2: Block diagram of the installation interactions.

We start building a list of spectators based on the depth and skeleton information, which has the following purposes: it contains the currently tracked spectators, identified by ID, ordered by who was first detected and various information in different times about tracked joints (see *skeleton* (Kinect, 2014b)). The list was also capable of knowing how many spectators had been near the installation for statistical purposes. Once a spectator was detected by Kinect this was added to the list alongside its current position, which will be used to check if he/she has moved and in what direction has he/she done so. In another hand, if that spectator or any other that was currently on the list stops being tracked by Kinect, he/she is removed from the list. If the list is empty, and a spectator was found, the initial film was played, at which it will never play again unless all active spectators exit Kinect field of view, shutting down the installation. Once this intro was played the installation will have a different reaction based on what the user did (Fig. 2.2).

As already mentioned in a previous Section, the spectator interacts with the installation based on his/her position relative to its previous tracked position and it also takes into account movement of his/her hands. Continually, the first spectator current position was compared to its last position, to know if and in what direction has he/she moved. Was considered that a spectator has moved if he/she was at least 0.5 meters away from its last tracked position, verifying then in what direction has he/she moved. For this we use the values returned by Kinect, that returns a position based on a three axis coordinates system $(x, y, z)$, where $x$ was the distance in meters the spectator was from the Kinect in the horizontal plane in a perpendicular direction of where Kinect was pointed, $y$ was the distance in meters the spectator was in a vertical plane from the Kinect and $z$ was the distance in meters that the user was from the Kinect in the horizontal plane in the direction that Kinect was pointed (for more details see e.g. Kinect (2014b)).

To know in which direction a spectator has moved the following procedure was implemented: (a) Obtain the current position of a spectator, (b) compare it with the

last tracked position of that same spectator, and verify if the absolute difference of each coordinate was greater than 0.5 m. If that comparison is true for a coordinate, then the spectator has moved in that axis triggering the next step. (c) A spectator has moved left if the subtraction between the current $x$ position and the last tracked $x$ position is negative, otherwise he has moved right. A user has moved up if the subtraction between the current y position and the last tracked $y$ position is positive, else that user has moved down. The same for $z$ position, being negative for forward, positive for backwards. (d) Once this was computed, the spectator joints (see Kinect (2014b)) and skeleton positions are updated (tracked) to be used in the next loop for the next comparison. Depending on what happened, the installation triggers a different reaction (see Fig. 2.2).

Two more cases must be stressed: First, in case the spectator has moved closer to the installation, his/her face will be extracted. For this Kinect SDK (Kinect, 2014b) does most of the job. (i) We pass to the SDK the spectator ID we want to extract the face, in which we will get a points cloud regarding to various points of the face. These points were returned by the Kinect, which includes various points of the contour of the extracted face among other points such as mouth, eyes, nose and eyebrows. (ii) With the contours point was built a Boolean polygon that was used as a mask to extract the face by simply applying it to the RGB image returned by Kinect. (iii) Both the mask and the RGB image were then cropped in order to obtain smaller images that are easier and faster to work, that were afterwards (iv) brightened up to match Our Lady "skin tone", which was done by adding up a constant to each pixel of the RGB image, and resized to fit Our Lady face. (v) Finally, the face can now be placed over Our Lady using a pre-known location of the image.

The second, was how to interact with the installation using gestures. In this case, a praying gesture (can be any gesture) was triggered if a spectator joins his/her hands at the level of the shoulders. To detect this, we (i) check in the spectator skeleton in the top of the list for 2 conditions: (ii.1) If the spectator hands are closer than 10 cm

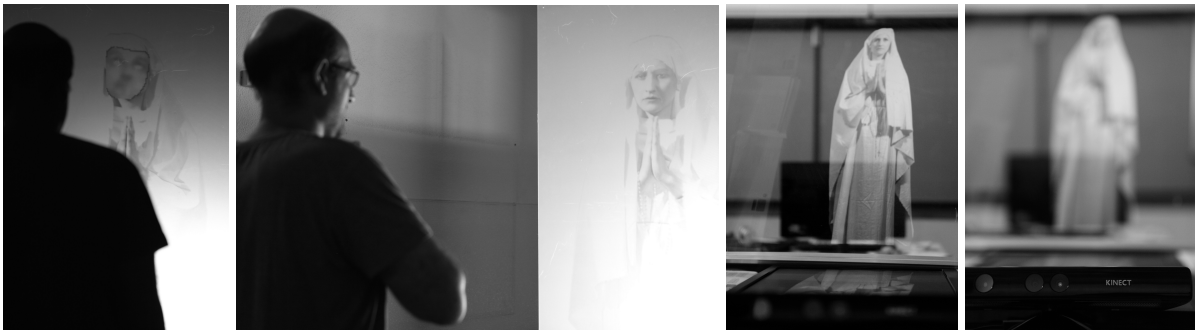


Figure 2.3: Examples of the Rear Projection (left) and Holographic installation (right).

of each other and (ii.2) if the hands are closer than 30 cm from the shoulders centre. If this was the case, (iii) is incremented a frame counter by a constant $k$ ($k = 2$) otherwise that same counter is decremented ($k = -1$). This counter will reflect how many frames the spectator have actually been performing the gesture. Note that $k$ cannot be lesser than 0, and if the $k > 40$, than the spectator was considered to be performing the praying gesture. Fig. 2.3 shows some examples of the two installations working. On the left the Rear Projection installation, and on the right the holographic installation using the movie built on the basis of the rightmost image in Fig. 2.1. The images are shown in grey to focus the attention of the reader to the installation not to surrounding background.

## 2.7 Discussion and Results

We presented two installations that use a Microsoft Kinect sensor to allow user interaction with a holy figure representing *Our Lady of Fátima*, with both installations following the same storyboard. We argue that the "mystic" effect created by both installations was able to tap into the Portuguese folklore and imaginary surrounding this chapter of Portuguese religious history. Both installations so far were only presented in the University Campus, nevertheless it was concluded that despite the holographic installation attracting more curiosity from the public, spectators preferred to interact with the Rear Projection installation – this was expected due to the different dimen-

sions of the installations.

In the future, with the final version of the holographic installation (not the prototype shown here), both installations will be presented at the same time in the same environment for a long period (around one week), and a more comprehensive evaluation, using specialized frameworks to evaluate public installations (Mathew et al., 2011). More conclusion will be taken from the different impacts of the installations, with a follow-up interview to each spectator.

# 3

# PRHOLO: Interactive Holographic Public Relations

## 3.1 Abstract

The expansion from regional to global markets is nowadays a normal step in the life cycle of many successful companies. Public relations are company's first contact with potential clients, requiring them to give a small explanation but yet accurately enough on what the company does and how the client can benefit from it. By combining a three-dimensional depth sensor with a holographic representation, an installation was developed capable of natural interaction with users, which can be configured for the

company's needs, while passing important information. Users are capable of interacting with the holographic form using intuitive gestures and the system is capable to record users' interactions, creating key statistics to return to the company about their main products, the attention given to each presented content, user's actions, favourite menus, etc., including the creation of a personalized visitor cards with the users face.

## 3.2 Introduction

A public relations has the fundamental job, within a company, to spread information to potential and regular clients. For this reason, it is very important that public relations (PR) know the right way to capture the attention and give good first impression to potential clients, persuading them that the company has the right tools and products to fit their needs. A different approach from the human PR is a holographic PR, which can capture the attention of a client using manifold visual (and sound) effects, and at the same time allowing a natural interaction between the user and the computer. Projections techniques are an interesting way to achieve it, since the created virtual character is not limited to screen size and can have a real size. On the other hand, holography (Antonio et al., 2013; Mihaylova, 2013) is one of the most realistic form of image and video display and can sometimes, be defined as a photographic image that is two-dimensional but appears to have depth. Pepper's Ghost (Sprott, 2006) is one of the most popular techniques to create a holographic illusion and can be combined with projection. This last case is implemented using a projector with a retention film, to hold the projected image, while an acrylic sheet (or Mylar foil) reflects that image to the spectator, which is placed at a 45º (degrees) angle to the retention film. The combined result gives the illusion of an image floating in the air while also giving depth perception; see e.g. D'Strict (2014); Moser (2014).

On the other hand, one of the new paradigms for Human-Computer Interaction (HCI) are the three-dimensional sensors, such as the Microsoft Kinect (Kinect, 2014b),

the Asus Xtion (Asus, 2014), the Leap Motion (Motion, 2014) or the Structure Sensor (Structure, 2014). Those sensors can be used for instance to interpret specific human gestures, enabling a completely hands-free control of devices. Hence, with the appropriate software, most of them have also the capability to detect the user's skeleton and/or tracking a single or several users, while replicating with accuracy the hands and the user movements in a 3D mesh. One of the most well-known 3D sensors is the Kinect, due to the game industry. Nevertheless, there are many other applications where this sensor is used, as for instance enabling interaction with art installations (Alves et al., 2014), in robotics (El-laithy et al., 2012), for head pose classification (Yun et al., 2014), applied in assistive technologies, as the enhancing visual performance skills on older adults (Chiang et al., 2012) or for the operation of wheelchairs (Kondori et al., 2014). More challenges and applications of the Kinect can be found e.g. in Cruz et al. (2012). Of course, HCI can be done with other 3D sensors, such as the mentioned Leap Motion, an example can be found in Figueiredo et al. (2014a) where the interaction is done with holograms for teaching technical drawing.

In this Chapter an interactive human size holographic PR from an industrial (commercial) installation is presented. The hologram can be represented by an avatar or by a video from a real human PR, both allowing to show different contents (text, image, video, maps, etc.). The interactivity is achieved using a Kinect sensor and very intuitive gestures, recording all the users' interactions, creating key statistics to return to the company about their main products, the attention given to each presented contents, users' actions, favourite menus, viewing direction, face and body extraction, etc.

Despite the already huge amount of applications that uses HCI, as far as we know none have all this characteristics mentioned above. The main contribution of the Chapter is real size customizable interactive PR installation, that benefits from a natural interaction with the user, granting more realistic human (hologram) to human interaction, while building statistics about users interactions.

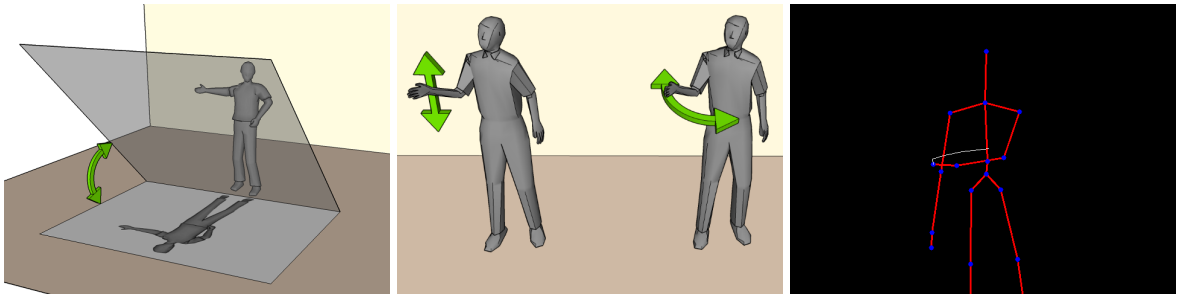

Figure 3.1: On the left, implemented holographic scheme using Pepper's Ghost technique. On the centre, implemented gestures. On the right a swipe gesture being performed.

## 3.3 PRHOLO system

The installation is divided in three main modules: The Users Data (a), the Kinect extracts spatial coordinates from users joints and colour image, to detect the interaction, create statistics and any other relevant information (e.g., user body and face image). The Database (b) is responsible for storing menus options and configurations (collected from the Backoffice), as well as storing statistics collected from (a). The Interface (c) is responsible for processing visual information, displaying menus and the hologram. For the hologram projection, the Pepper's Ghost technique was used. Fig. 3.1 left shows the basic illustration of the technique; a detailed explanation is out of the focus of the Chapter, for more information please refer to Antonio et al. (2013); Mihaylova (2013); Sprott (2006); D'Strict (2014); Moser (2014); Figueiredo et al. (2014a). In the following sections all three modules are explained in more detail.

### 3.3.1 Users Data Module

Users Data module is responsible for handling and manipulating received data from Kinect (not Kinect 2). This sensor provides colour (RGB) and depth frames, 25 joints of 2 users and can track up to 6 users (not used here, it can also provide sound); for more details see Kinect (2014b). The above data is manipulated in 5 main phases: (a) spatial information, (b) gestures recognition, (c) heat map, (d) user head direction, (e)

body and face extraction:

**(a) Spatial Information** $(x, y, z)$ of user's joints including the global position ($P$), are stored using two FIFO (First In, First Out) list, one for the current (detected) users ($Pc$) and another for lost users ($Pl$). Every time the Kinect has new information available about users, 3 different situations may occur: (i) new users are available, which are added to the end of the current users list ($Pc$) alongside all joints information and entry time ($t$), but (ii) if the user already exists, then the new information is stored with the previously obtained information on the current users list, alongside the entry time of the new information. (iii) If a user is lost, then all information is updated in the database and the user is discarded from the current users list and added to the lost users list. Since users can sometimes block the Kinect's view, a simple occlusion detection was built to recover a lost user. When a new user is detected, it is verified if, in the lost users list, there are lost users less than $\Delta t$ seconds ago. For all of the last positions of all lost users (i) in the past $\Delta t$ seconds, $Pl_i$, the current position of a detected user (u), $Pc_u$, is considered to belong to a previous user if the Euclidean distance (d) $d_{i,u}$ between $Pl_i$ and $Pc_u$ is less than 30 cm. If more than one lost user is closer than 30 cm to the position $P_u$ then the closest distance $d$ is chosen, recovering all information to the current users list (it was used $\Delta t = 5$s).

**(b) Gestures recognition** is responsible for inputs detection from users. After experimenting with different gestures, the "swipe gesture" and the "pose gesture" were chosen and implemented due to their intuitive nature for the users; Fig. 3.1 centre, illustrates those gestures, in the left the "pose" and on the right side of the image the "swipe". For the swipe gesture implementation, given minimum swipe distance $d_s$ and the minimum swipe velocity $v_s$, a time window $\Delta t_s$ can be calculated with $\Delta t_s = d_s / v_s$. By analysing only user information acquired from current instance $t$ to $t - \Delta t_s$, a swipe was made if in any other sub-interval $\Delta t_k$, defined between $t$ and $t - \Delta t_s$, with $k$ the latest instance of $\Delta t_k$, all hand positions ($h$) minus its previous hand position, taking only into account the $x$ component for the horizontal swipe, or

the $y$ component for the vertical swipe, has the same signal along the whole interval $\Delta t_k$, $|\sum_{j=k-\Delta t_k}^{k-1} sgn(h_{j+1,\{x/y\}} - h_{j,\{x/y\}})| = k-1$, and if the total distance travelled, taking only into account the $x/y$ component respectively for the horizontal and vertical swipe, between the first and the last point of that sequence is greater or equals to $d_s$, that is, $d_s \geq |h_{k,\{x/y\}} - h_{k-\Delta t_k,\{x/y\}}|$.

Fig. 3.1 right, illustrates a horizontal swipe. The minimum distance used was $d_s = 30$ cm and the minimum speed used was $v_s = 200$ cm/s. The pose gesture can be detected using the vector defined by the user's body $\vec{V_b} = (x_b, y_b, z_b)$, which is computed by subtracting the shoulder centre position, $P_{sc}$, from the spine position, $P_s$, and using a vector defined by the arm, $\vec{V_a} = (x_a, y_a, z_a)$, computed by subtracting hand (right / left) position, $P_h$, from the elbow (right / left) position, $P_e$. The user is performing the menu pose if the two following conditions are true: (1) As the user must be facing Kinect, the distance between the elbow and the Kinect must be approximately the same as the hand and the Kinect. For this reason, only if $|z_h - z_e| \geq 0.6 \times \mathrm{d}(P_h, P_e)$ is considered that the user is performing the gesture. (2) An angle $\theta$ between $\vec{V_b}$ and $\vec{V_a}$ can be measured with $\theta = \arccos(\vec{V_b} \times \vec{V_a}) / (|\vec{V_b}||\vec{V_a}|)$. If the angle is $20^o \leq \theta \leq 160^o$, then the user is doing the pose gesture. To increase the reliability of the pose gesture, it is verified if at least 85% out of the previous detections made in the past 1 second are according to the above conditions. If the user is performing the pose gesture, it is considered - up - if the angle $\theta$ is less than $90^o$, and - down - otherwise.

**(c) Heat Map** of users is also calculated, useful for statistics about the most active locations of a user or a group of users. The users' global position ($P$) returned by Kinect, Fig. 3.2 first row, represents the distance in each axis between the user and the Kinect, with the $x$ axis representing the horizontal distance and the $z$ axis the vertical distance. In this application, it is assumed that the physical limits of the Kinect is 4 m in length and in width, (Kinect, 2014b). A matrix $M$ with size of $N \times N$ can be created, dividing the map in squares of approximately $(4/N)$ m $\times (4/N)$ m. In the present case, it was considered $N = 26$, consequently each square as the area of

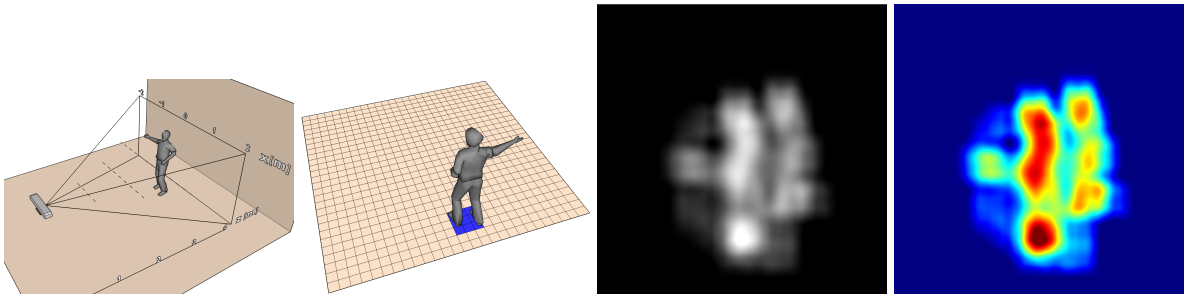

Figure 3.2: From left top right, Kinect coordinate system,a user being mapped to matrix $M$, the normalized matrix $M$ and the same matrix converted to the JET colour map.

0.024 m$^2$. A user position detected from Kinect on instance $t$ is mapped to matrix $M$ using $(x_t, y_t) = (-x_{k_t} \times (N/4) + N/2, z_{k_t} \times (N/4))$, with $x_k$ and $z_k$ the coordinates obtained from Kinect (Fig. 3.2 second row). Starting with every position of the matrix $M$ equals to zero, every detection made at instance $t$ increments its value, as well as its 8 neighbours: $M(x_t + i, y_t + j) = M(x_{t-1} + i, y_{t-1} + j) + 1$, with $i$ and $j = \{-1, 0, 1\}$. After this step, the matrix $M$ holds values of the positions that a user, or a group of users (optional). The matrix is then normalized, Fig. 3.2 third row, and converted to a colour map using JET colour map, visible on same Figure fourth row.

**(d) User Head Direction** is used to estimate in which direction the user was looking and the time spent looking at the installation while interacting. Since Kinect SDK only allows two people for face tracking, only the first two users from the current users list are analysed. Getting a precise point where a user is looking to is difficult, mostly due to Kinect colour resolution. For this reason, reference points are used for the user's head to determine where the person is probably looking to, despite a person's head might be facing a direction while his/her eyes may be looking elsewhere. Using Kinect SDK the position of the users' eyes are obtained, in the colour frame, (Fig. 3.3 red dots). This information is used, alongside the shoulders $(S)$ position (left, centre and right; black dots in the same Figs) where he/she might be facing. With the coordinates of the points, it is calculated the normal vector $\vec{V}_s'$ to the vector $\vec{V}_s$, defining a line containing both shoulder left and right points in the colour frame. Vector

Figure 3.3: On the left, user looking to his right. On the centre, user looking to the middle. On the right, user looking to his left.

$\vec{V_s}$ is calculated using shoulder left $(l)$ and right $(r)$ points, $S_{l/r} = (x_{S_l/S_r}, y_{S_l/S_r})$, with $\vec{V_s} = (x_{S_l} - x_{S_r}, y_{S_l} - y_{S_r})$ thus being $\vec{V_s'} = (-V_{S_y}, V_{S_x})$. Applying $\vec{V_s'}$ to the shoulder centre $(c)$ Position (vertical or almost vertical blue line in the same Figs), three different scenarios can happen: (1) a user can have each eye on each side of the line, Fig. 3.3 centre, meaning that the user is looking to the centre, (2) both eyes can be placed on the left side of the line, Fig. 3.3 left, thus the user is looking to the right and finally both eyes are placed on the right side of the line, Fig. 3.3 right, determining that the user is looking left. Depending on the position where the user is, extreme left and right directions were excluded. A timer was implemented to count the time spent looking in each direction.

**(e) Face and Body Extraction** is done and shown to the user as soon as he begins to interact with the installation. While the Kinect SDK does all of the job extracting the user face, Fig. 3.4 first image, full body photo is not done directly. Every time a new depth frame is available, the first three bits represents which user that pixel belongs to, from 1 to 6 and with 0 meaning it belongs to no user. A Boolean mask for each user is constructed, visible in Fig. 3.4 second image. For that image the value for the highest and lowest $x_{\{min/max\}}$ and $y_{\{min/max\}}$ is found. The colour image is then cropped $(U)$ using this points. An example can be seen in Fig. 3.4 third image. Alternatively a Gaussian filter is applied to the Boolean mask $(\sigma = 2)$, and again the image is cropped and a bitwise AND logic is performed with $U$, obtaining Fig. 3.4 fourth image. These images (face and body) are used to insert the user in different backgrounds/situations

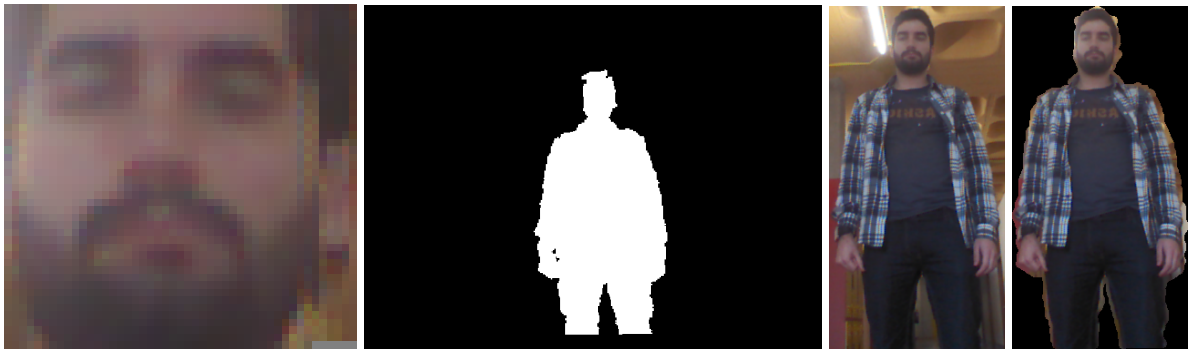

Figure 3.4: From left to right, extracted user's face, Boolean mask of the user, extracted user's body and extracted user's body with background removal.

and then offered to the user as "promotional gifts".

### 3.3.2 Database Module

The database module stores two types of information: Menus configurations (i) enabling customization of the whole application changing the appearance of it to match companies' schemes, colours and logos. To adapt the interface to the company's needs, the interface can be entirely changed: (a) Content of the application can be customized. An application can have several menus, each menu with any number of contents, with an action associated with it. A content action can be: opening another menu, running a video or displaying an image. (b) To fit different types of information, 6 different menu layouts were build. (c) Design of the menus can also be changed. Images of buttons, font type, font colour and background images can be changed within a menu. Once a menu has been created, a company can change any of those parameters. (d) The avatar can also be changed to any 3D model of a character, or a video of a human PR.

Statistics (ii) can be quite useful to know users feedback, which information has been most requested or the time spent on each content. The following information is stored: (a) Users spatial position, alongside joints positions, enabling a replication of what the user has done while interacting with the installation. (b) Menus statistics are also saved. For each user it is counted the time he/she spent on it and also the number

of selections. This is useful to determine which information is most requested by users. The user face and body photo are also saved and can be, afterwards, matched with the information requested to see what kind of product was a user most interested on.

### 3.3.3 Interface Module

The Interface module is responsible to read information stored in the database and automatically generate layouts, as well as generating avatar responses to the user input. To interact with the menus, in the present installation were used the horizontal swipe and the pose gestures (Section 3.3.1). In resume, the application can generate the following layouts: (a) titles of menu contents useful to serve as bridge to other menus, visible in Fig. 3.5 centre, (b) similar to (a), a layout that can fit small description if needed. In specific cases, (c) media content as image or video with description is useful to get a user the idea beyond a concept, generating a layout similar to (b). Alternatively, the same layout can be used but displayed in a diagonal instead of vertical. (d) One final option is available to display only images or video, Fig. 3.5 left. All configurations are loaded when the application starts. The avatar trigger responses to inputs of the user: (a) If the interacting user with the installation changes, the avatar waves while showing that user's photo. (b) If the user selects an option, the avatar touches its content triggering a menu change. It is important to accentuate that despite the images in Fig. 3.5 centre and right do not give that notion, the holographic installation has the height of $\approx 2$ m and a width of $\approx 1.2$ m and the lady avatar has the height of a human lady $\approx 1.7$ m.

## 3.4 Conclusion

In this Chapter a PR installation was presented, interacting more naturally with users while enhancing their visual experience. Using a holographic technique, an avatar can interact with a user giving him/her information about a company. Using a Microsoft


Figure 3.5: From left to right, extracted user's face, Boolean mask of the user, extracted user's body and extracted user's body with background removal.

Kinect, users are detected and can interact with the avatar using gestures, while tracking all information about them, useful for statistics. Statistics about users' positions and body joints are stored in a database, being useful for heat maps as well as statistics about time spent on each content and requested information.

The results of swipe gestures were good, as previous hand positions were taken into account. The detected successful rate was 98%, being too difficult to tell exactly what did fail. Usually it was due to the user not making the gesture correctly, due to a lack of speed or gesture extend (distance). In respect of user looking estimation, as expected, the results were not as good. If a user is facing towards the Kinect, the result obtained works as expected, giving an estimation on where a person might be looking, despite his/her eyes might be looking elsewhere, but if the user is sideways to Kinect results are not reliable enough, although it is not a critical error for this application.

Future work includes improving recognition of previous user, not only when Kinect view is blocked but also when an user returns some hours later to ask for more information, as well as improving the user head direction.

# 4

# Interactive 180° Rear Projection Public Relations

## 4.1 Abstract

In the globalized world, good products may not be enough to reach potential clients if creative marketing strategies are not well delineated. Public relations are also important when it comes to capture clients attention, making the first contact between them and companies products while being persuasive enough to trust that the company has the right products to fit their needs. A virtual public relations is purposed, combining technology and a human like public relations capable of interacting with potential

clients located $180^o$ (degrees) in front of the installation, by using gestures and sound. Four Microsoft Kinects were used to develop the $180^o$ model for interaction, which allows tracking and recognition of gestures, sound sources, voice commands, extract the face and body of the user and track users positions (heat map).

## 4.2 Introduction

Maintaining regular clients and capturing attention of new clients is one of the very first concerns a company deals with. Through well trained public relations (PR), companies reach their clients persuading that their products and tools are a major asset and can be beneficial to their business and problems. New clients are not often familiar with the company concept and have many questions on what products are available and what services can be provided to them, and normally a website can clear them up, but with many companies in the same working field, it might be difficult to excel them.

A real sized installation with stunning visual effects is more likely to capture a new client attention. There are several techniques for the projection of real size PR persons (movies) or avatars. Frontal projection is most common, but the worse solution, once the user can conceal the projection. Other techniques including Pepper's Ghost technique (see e.g. Figueiredo et al. (2014a)), requires a lot of space. Rear projection technique using a ultra short throw projector occupies a very small/limited space, and consists on a projector projecting onto a retention film, enabling a group of people to see what's projected on the other side of the retention film without them seeing the projector and with the advantage of creating an image not limited to size.

To develop the interaction with an installation, there are many sensors and cameras that can be used, nevertheless, three-dimensional (3D) sensors, such as the Microsoft Kinect (Kinect, 2014b), the Asus Xtion (Asus, 2014), the Leap Motion (Motion, 2014) or the Structure Sensor (Structure, 2014) are gaining popularity due to their capability of

capturing Natural Interaction (NI) enabling hands-free control of devices and menus, while tracking user's skeleton and recognizing joints and gestures of one or several users. One of the most well-known 3D sensors is Microsoft Kinect, due to the game industry.

There are many applications where this sensor is used, as for instance enabling interaction with art installations (Alves et al., 2014), in robotics (El-laithy et al., 2012), for head pose classification (Yun et al., 2014), applied in assistive technologies, as the enhancing visual performance skills on older adults (Chiang et al., 2012) or for the operation of wheelchairs (Kondori et al., 2014). More challenges and applications can be found e.g. in Cruz et al. (2012); Fong et al. (2013); Kamizono et al. (2014); Rahman et al. (2015). Of course, interaction can be done with other 3D sensors, such as the mentioned Leap Motion, an example can be found in Figueiredo et al. (2014a) where the interaction is done with holograms for teaching technical drawing.

In this Chapter a real-sized humanoid character using rear projection technique is presented. The main contribution is the developed model, which is capable of tracking users position, gestures and sound, for up to $180^{o}$ in form of the installation. By using 4 Microsoft Kinect sensors, all the users in front of the installation are tracked on-the-fly, and selected the user and respective sensor in which the gesture intersection will occur. In the case of the absence of a user in front of the installation, it is searched for the highest sound source, and the best suitable sensor is used to detect voice commands (words ou small sentences). Every interaction, tracking, information extracted from the users (e.g., biometric information) and requested to the installation is recorded in a database, creating statistical data (for the company analysis), such as users actions, favourite menus, etc. Although there are already a huge amount of applications that uses NI, as far as we know none have all the characteristics mentioned above.

## 4.3 Concept and Installation

As already mentioned, the main goal is to develop an interactive installation with rear projection to project a real size human figure plus menus, videos, etc. The installation consists, on present case, of a transparent acrylic (to optimize cost) with a retention film and an ultra throw short projector to minimize distance from the retention film. Fig. 4.1 top left, shows the physical installation layout, i.e., the projector, with the projection surface in front, and the sensors in front of the surface. In the present installation the sensors were almost in the ground, but the sensor group can be moved in the vertical to any position, provided it meets the layout.

The application is divided in 4 modules: (a) Users Data, responsible for handling and manipulating information received from each individual sensor Kinect (S0 to S3), reading gestures, sound and creating statistics, (b) Global Management, responsible to convert spatial information of users from each individual sensor to a global reference and disambiguate users that were detected by multiple Kinects. (c) The Database is responsible for storing menu options and configurations (collected from the Backoffice), as well as storing information collected from (a) and (b). (d) The Interface is responsible for processing visual information, displaying menus and the virtual character. The modules (a) and (b) consists in the $180^o$ model for NI, the development of modules (c) and (d) is out of the focus of this Chapter, nevertheless they will be very briefly explained for system comprehension purpose in Section 4.3.3.

The interaction model, consists in 4 Microsoft Kinect sensors, to capture a total field of view of $180^o$, positioned in a way that maximizes overlap areas of neighbour Kinects (see Fig. 4.1 top left, for the layout illustration). For this, three physical properties of Kinects are needed: (a) their position $C_l(x,y,z)$ in a global reference (the red dot in Fig. 4.1 bottom left is the origin), with $l$ being the Kinect index, $l = \{0,..,3\}$, which represents Kinect positions relative to each other, (b) their horizontal rotation $\beta_l$ and (c) their vertical angle $\phi_l$ with both representing the direction each one of them are facing.

Note that the vertical angle doesn't need to be physically calculated, since Kinect SDK has methods that returns Kinect vertical angle, taking advantage of Kinect's built in accelerometer.

To determine the Kinect' position $C_l$ it is necessary to calculate first the horizontal angle they are facing. Since the installation captures $180^o$, visible in Fig. 4.1 second left row, their intersection angle $\alpha$ can be calculate with $180 = 4 \times \lambda - 3 \times \alpha$, with $\lambda$ being Kinect's horizontal field of view angle $\lambda = 57^o$ (Kinect, 2014b), returning an angle $\alpha = 16^o$. The horizontal angles $\beta_l$ can now be calculated, starting with the Kinect on the right and counter-clockwise, Fig. 4.1 left second row: $\beta_0 = \lambda/2 = 28.5^o$ and $\beta_i = \beta_{i-1} + \lambda - \alpha$, with $i = \{1,...,l\}$, returning $\beta_1 = 69.5^o$, $\beta_2 = 110.5^o$ and $\beta_3 = 151.5^o$.

Ideally, positions $C_l(x,y,z)$ would all be the same for all Kinects but this is not possible, since Kinect has a width of $w_{KS} = 28$ cm (centimetres) (Kinect, 2014b), making it impossible to be placed all in the same point. Due to this, all Kinects need to be distanced of this point by a distance of $r$ in the direction of the angle they are facing, visible on Fig. 4.1 left second row, and can be calculated with $r = w_{KS}/(2 \times \tan(\omega))$ and $\omega = (\beta_1 - \beta_0)/2$, giving an $r \approx 36$ cm (centimetres). The positions $C_l$ can be calculated with $C_l(x,y,z) = (r \times cos(\beta_l), 0, r \times \sin(\beta_l))$, thus $C_0(x,y,z) \approx (32,0,17)$ cm, $C_1(x,y,z) \approx (12,0,34)$ cm, $C_2(x,y,z) \approx (-12,0,34)$ cm and $C_3(x,y,z) \approx (-32,0,17)$ cm. Having the physical positions of the Kinect sensors, it is now possible to compute the remaining components of the NI model.

### 4.3.1 Users Data Module

As stated in the previous Section, this module is responsible for handling and manipulating received data from each single Kinect, with all steps described in this Section being replicated for each of the 4 Kinect sensors used. Each sensor provides sound extraction, colour (RGB) and depth frames, 25 joints of 2 users and can track up to 6 users. The above data is manipulated in 4 main phases: (a) spatial information, (b) gestures recognition, (c) body and face extraction and (d) sound extraction.

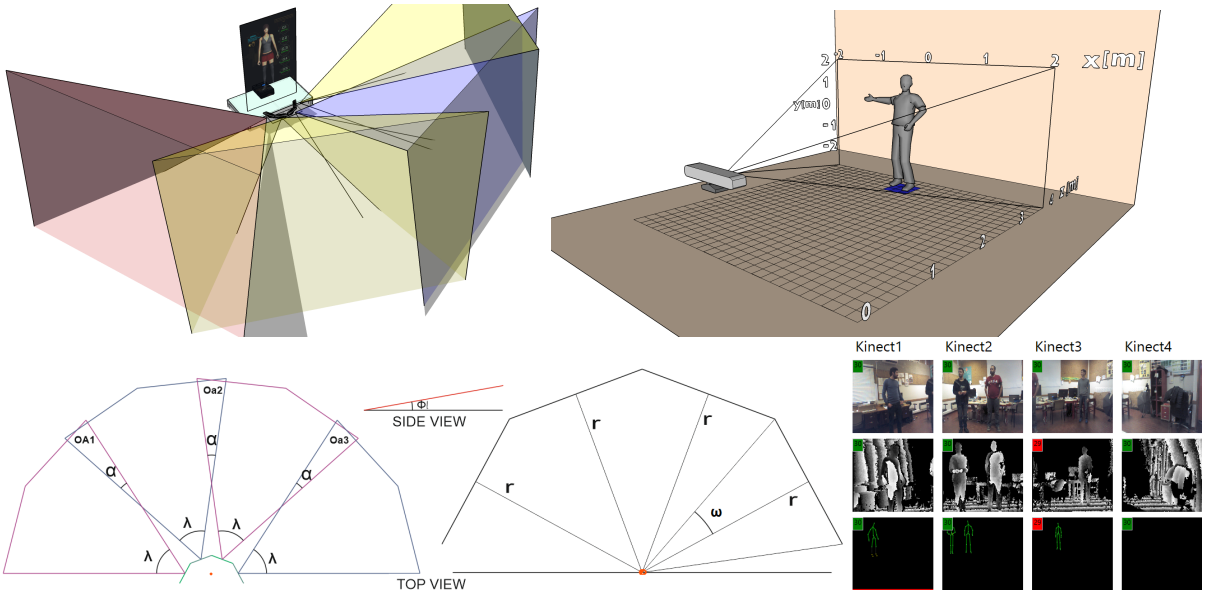


Figure 4.1: Left to right, and top to bottom: the physical installation layout, Kinect coordinate system, the physical coordinates and angles relative to each Kinects and a real time data streaming from 4 Kinects: RGB, depth and skeletons.

### Spatial Information

Spatial Information $(x, y, z)$ of user's joints including the global position $(P)$, are stored using two FIFO (First In, First Out) lists, one for the current (detected) users $(Pc_l)$ and another for lost users $(Pl_l)$ (as mentioned, $l$ is the Kinect number). Every time the Kinect has new information available about users, 3 different situations may occur: (i) *new users are available*, they are added to the end of the current users list $(Pc_l)$ alongside all joints informations and entry time $(t)$ and assigned an internal ID, which is then incremented, but (ii) *if the user already exists*, then the new information is stored with the previously obtained information on the current users list, alongside the entry time of the new information. (iii) *If the user is lost*, then all information is updated in the database and the user is discarded from the current users list and added to the lost users list.

**Gestures Recognition**

Gestures recognition is responsible for ("the visual") inputs from the users. After experimenting with different gestures, the "swipe gesture" and the "pose gesture" were chosen and implemented due to their intuitive nature for the users. Fig. 4.2 first column, illustrates those gestures, in the left the "pose" and on the right side of the image the "swipe". These gestures were chosen to be somewhat intuitive while not being too frustrating.

For the ***swipe gesture*** implementation, given minimum swipe distance $d_s$ and the minimum swipe velocity $v_s$, a time window $\Delta t_s$ can be calculated with $\Delta t_s = d_s / v_s$. By analysing only user information acquired from current instance $t$ to $t - \Delta t_s$, a swipe was made if in any other sub-interval $\Delta t_k$, defined between $t$ and $t - \Delta t_s$, with $k$ the latest instance of $\Delta t_k$, all hand positions ($h$) minus its previous hand position, taking only into account the $x$ component for the horizontal swipe, or the $y$ component for the vertical swipe, has the same signal along the whole interval $\Delta t_k$, ($|\sum_{j=k-\Delta t_k}^{k-1} \text{sgn}(h_{j+1,\{x/y\}} - h_{j,\{x/y\}})| = k - 1$), and if the total distance travelled, taking only into account the $x/y$ component respectively for the horizontal and vertical swipe, between the first and the last point of that sequence is greater or equals to $d_s$, that is, $d_s \geq |h_{k,\{x/y\}} - h_{k-\Delta t_k,\{x/y\}}|$. Fig. 4.2 second column, illustrates an horizontal swipe. The minimum distance used was $d_s = 30$ cm and the minimum speed used was $v_s = 200$ cm/s.

The ***pose gesture*** can be detected using the vector defined by the user's body $\overrightarrow{V_b} = (x_b, y_b, z_b)$, which is computed by subtracting the *shoulder centre* (Kinect, 2014b) position, $P_{sc}$, from the *spine* position, $P_s$, and using a vector defined by the arm, $\overrightarrow{V_a} = (x_a, y_a, z_a)$, computed by subtracting the *hand (right/left)* position, $P_h$, from the *elbow (right/left)* position, $P_e$. The user is performing the menu pose if the two following conditions are true: (1) As the user must be facing the Kinect, the distance between the *elbow* and the Kinect must be approximately the same as the *hand* and the Kinect. For this reason, only if $|z_h - z_e| \geq 0.6 \times \text{d}(P_h, P_e)$ is considered that the user is performing the gesture, with d the Euclidean distance. (2) An angle $\theta$ between $\overrightarrow{V_b}$ and $\overrightarrow{V_a}$ can

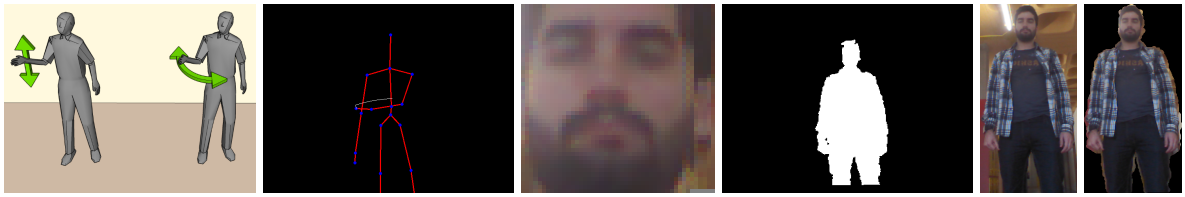

Figure 4.2: From left to right, the "swipe" and "pose" gestures, a "swipe" gesture being performed, as viewed from a single Kinect sensor, the extracted face of current user, the boolean mask of detected user, the cropped image of a user and the extracted user with no background.

be measured with $\theta = acos((\overrightarrow{V_b} \cdot \overrightarrow{V_a})/(|\overrightarrow{V_b}||\overrightarrow{V_a}|))$. If the angle is $20^o \leq \theta \leq 160^o$, than the user is doing the pose gesture. To increase the reliability of the pose gesture, it is verified if at least 85% out of the previous detections made in the past 1 s (second) are according to the above conditions. If the user is performing the pose gesture, it is considered - up - if the angle $\theta$ is less than $90^o$, and - down - otherwise.

**Face and Body Extraction**

Face and Body extraction is done and shown to the user as soon as he begins to interact with the installation. While the Kinect SDK does all of the job extracting the user face, Fig. 4.2 third column, full body photo is not done directly. Every time a new depth frame is available, the first three bits represent which user that pixel belongs to, from 1 to 6 and with 0 meaning it belongs to no user. A boolean mask for each user is constructed, visible in Fig. 4.2 fourth column. For that image the value for the highest and lowest $x_{\{min/max\}}$ and $y_{\{min/max\}}$ is found. The colour image is then cropped (U) using this points. An example can be seen in Fig. 4.2 fifth column. Alternatively a Gaussian filter is applied to the boolean mask ($\sigma = 2$), and again the image is cropped and a bitwise AND logic is performed with U, obtaining Fig. 4.2 sixth column. These images (face and body) are used to insert the user in different backgrounds/situations and then offered to the user as "promotional gifts".

#### Sound Extraction

The sound extraction for each sensor is done using the Kinect SDK, each Kinect is equipped with a built in microphone array, composed by 4 microphones. The microphone array is capable of sound localization also called beam forming, that is the ability to estimate the audio source direction by calculating differences between audio streams captured. The beam angle range covers $100^o$ in front of the Kinect, $50^o$ from centre to each side (Kinect, 2014b). The beam direction is an integer value multiple of 10, in all containing 11 different values. These values represent sound source direction, in addition the Kinect SDK estimates the confidence level that the audio source was located, represented by a value between 0 and 1. Speech recognition is also supported by the API, having only to provide words or small sentences that will be recognized.

### 4.3.2 Global Management Module

Global Management module is responsible for managing users and sound detected by each Kinect, creating a global reference, selecting the user that is interacting (gesture / sound), and solving the problem where a user is detected by two neighbour Kinects due to overlap on their field of view. The Global Management works similar to Users Data Module, Section 4.3.1, consisting in two FIFO lists, one for current detected users $Gc$ and other for lost users $Gl$.

#### Global Reference Conversion

Once all the Users Data modules are updated, all users from $Pc_l$, are transformed to a global reference and put into an array of potential users $Uc$. This conversion transforms the coordinates returned by Kinect (Fig. 4.1 top right), to a global reference, visible in Fig. 4.3 left first row, being the lighter area the interaction zone with a radius of 4 m (meters) counting from the centre of the Kinects setup. Each user is represented by a circle, occupying 50 centimetres. To convert a Kinect coordinate ($x$ axis is repre-

sented from left to right, the $y$ axis from up to down and the $z$ axis the depth distance, Fig. 4.1) to global reference, the physical layout of all Kinect cameras are necessary, keeping in mind their distance between each other, as well as their rotation in the horizontal plane $\beta_l$ and rotation on the vertical plane $\phi_l$, see Section 4.3.

The first step consists in finding the directive unitary vectors of each Kinect sensor, one for each axis: $\overrightarrow{V_a}$, representing the vector of the axis a Kinect is facing, $\overrightarrow{V_b}$, representing the axis with an horizontal $90^o$ to $\overrightarrow{V_a}$ and $\overrightarrow{V_c}$, representing the third axis with $90^o$ degrees to both $\overrightarrow{V_a}$ and $\overrightarrow{V_b}$ pointing upwards in relation to Kinect sensor. $\overrightarrow{V_a}$ can be found with $\overrightarrow{V_a}(x,y,z) = (\cos(\beta_l) \times \sin(\frac{\pi}{2} + \phi_l), \cos(\frac{\pi}{2} + \phi_l), \sin(\beta_l) \times \sin(\frac{\pi}{2} + \phi_l))$ and $\overrightarrow{V_b}$ being normal in the horizontal plane to $\overrightarrow{V_a}$ and calculated as $\overrightarrow{V_b}(x,y,z) = (\cos(\beta_l + \frac{\pi}{2}) \times \sin(\frac{\pi}{2} + \phi_l), \cos(\frac{\pi}{2} + \phi_l), \sin(\beta_l + \frac{\pi}{2}) \times \sin(\frac{\pi}{2} + \phi_l))$. Finally, $\overrightarrow{V_c}$ is calculates with $\overrightarrow{V_c}(x,y,z) = (\cos(\beta_l) \times \sin(\phi_l), \cos(\phi_l), \sin(\beta_l) \times \sin(\phi_l))$. Given that the Kinect SDK gives the coordinates in meters $\hat{J}$, a joint position $J$ of a user made by Kinect $l$ in centimetres is calculated with by $J_{x,y,z} = \hat{J}_{x,y,z} \times 100$. The new position in the global reference $Jg$ can now be calculated with $Jg(x,y,z) = (J_x \times \overrightarrow{V_b} + J_y \times \overrightarrow{V_c} + J_z \times \overrightarrow{V_a} + C_l)/100$ m, with $C_l$ being the global position of the Kinect. Now $Jg$ can represent the conversion of any joint or a user global position from any sensor to the global reference, by using this transformation we denote $Pg$ as the global position of the user in the global reference. All users are then mapped on the same reference, with an aerial view visible in Fig. 4.3 top left.

**Recognition of Multiple Kinects Detections of an User**

Since all Kinects overlap their field of view with their neighbours (see Fig. 4.1 top left), a method was developed to determine if a user was detected by multiple (2) Kinects. All potential users on $Uc$ are compared with each other to find if they might be the same user, using two criteria: (a) (a.1) if the euclidean distance between global positions of the two users are compared in $Uc$, for instance $Pg_1$ and $Pg_2$, is less then 50 cm and (a.2) the two users were detected by different Kinects, then (a.3) if the last

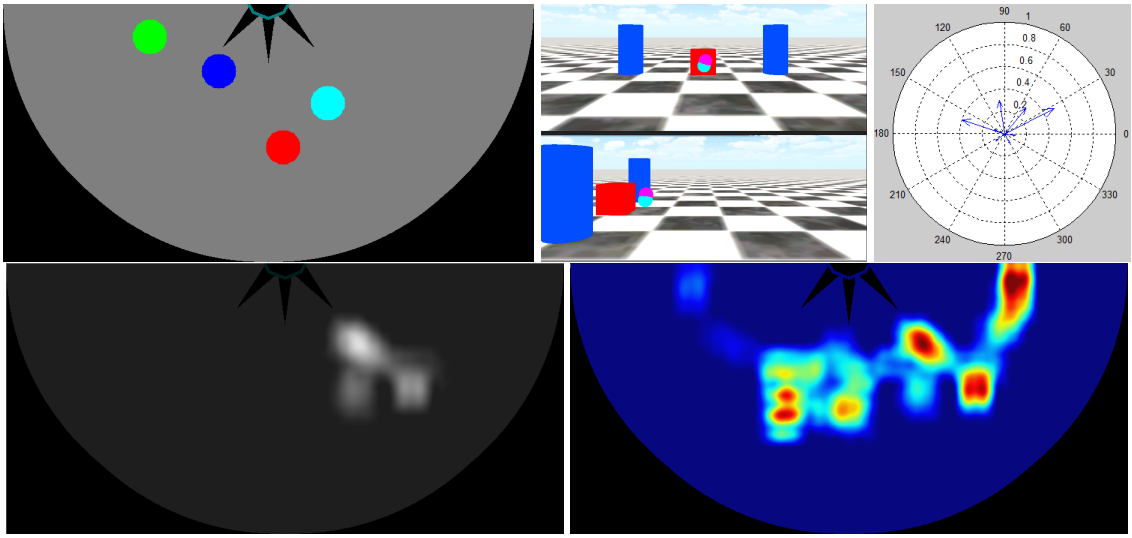


Figure 4.3: Top to bottom, left to right: an aerial view with 4 detected users, 2 different 3D views of the same user being detected by two different Kinects. A real time sound source beams from all 4 Kinects, in a polar plot, and a Heat Map of a single user, and the Heat Map from multiple user, on JET colour map.

$T$ skeletons information are available, with $T = 5$, it is compared if the global position variation on all axis $(x,y,z)$ are similar: $(|Pg_{1/2,x}(\delta) - Pg_{1/2,x}(\delta - 1)| < 15 \text{ cm}) \wedge (|Pg_{1/2,y}(\delta) - Pg_{1/2,y}(\delta - 1)| < 15 \text{ cm}) \wedge (|Pg_{1/2,z}(\delta) - Pg_{1/2,z}(\delta - 1)| < 15 \text{ cm})$, with $\delta = \{t,...,t - T\}$, then it is assumed that both users are in fact the same user. Finally, (a.4) if the relation between the users was found, the newest user will be forced to change his/her internal ID to the internal ID of the oldest detected user.

(b) In case the two users already share the same internal ID, then they are automatically related as the same user meaning that the relation was already found in a previous situation by method (a). In both (a) and (b) cases, the user closer to its Kinect will be marked for addition to the end of the list $Gc$, with the other discarded of that list. Fig. 4.3 top middle illustrates, using 2 views, the situation where two sensors detected the same user, with both converted to the global reference, the solids in the image were only used to get a better comprehension of the 3D space.

### Updating, Adding and Removing Users

Once the previous step (Section 4.3.2) is complete, all marked users will be added to the current users list $Gc$, with a detected user being either completely new to this list or the user being new information of an already existing user of $Gc$. For all users marked for addition on $Uc$, with the user being $u_i$ and $i$ ranging from 0 to the size of array $Uc$, are compared to each of the users already on $Gc$, with the user being $gc_j$ and $j$ ranging from 0 to the size of list $Gc$. If both users $u_i$ and $gc_j$ share the same internal ID, then the user $gc_j$ will be updated with the last skeleton (on global reference) of the user $u_i$. In case user $u_i$ does not share the same internal ID with any user of $Gc$ then the user $u_i$ is new and is added to the end of the list $Gc$. On the other hand, if a user $gc_j$ is not sharing the same internal ID with any marked user of $Uc$, then user $gc_j$ is discarded from $Gc$ list and added to the end of the list $Gl$.

Since users can sometimes block the Kinect's view, a simple occlusion detection was built to recover a lost user. When a new user is detected, it is verified if, in the lost users list $Gl$, there are lost users less than $\Delta t$ seconds ago. For all of the last positions of all lost users ($k$) in the past $\Delta t$ seconds, $Gl_k$, the current position of a detected user ($u$), $Gc_u$, is considered to belong to a previous user if the Euclidean distance $d_{i,u}$ between $Gl_i$ and $Gc_u$ is less than 50 cm. If more than one lost user is closer than 50 cm to the position $P_u$ then the closest distance d is chosen, recovering all information to the current users list $Gc$. It was used $\Delta t = 5$ seconds.

The *selected user* to interact with the installation is the one in the $Gc$ list that is closer to the installation, and remains the *selected user* until this leaves the $180^o$ space analysed by the system. If the user leaves the space or does not have any type of movement during 30 s then the *selected user* is the next in the $Gc$ list closest to the installation.

### $180^o$ Heat Map

One important feature of the model is the Heat Map of users, very useful for statistics about the most active locations (and time spend at that location) of a user or a group

of users. By using the users global position ($Pg$) (see Section 4.3.2) the Heat Map can be calculated. In this application it is assumed that the physical limits of the Kinect is approximately 4 m in length and in width (Kinect, 2014b) (in fact it is a bit less), and thus the heat map generated represents a region of approximately $8 \times 4$ m. In reality it is more or less a circle of 4 m radius considered from the middle of the Kinect group (see Figs 4.1 and 4.3 top left). A matrix $HM$ with size of $2N \times N$ can be created, dividing the map in squares of approximately $(4/N) \times (4/N)$ m. In the present case, it was considered $N = 26$ px (pixels), consequently each square as the area of 0.024 $m^2$. A user's position obtained after coordinate transformation (Section 4.3.2) on instance $t$ is mapped to matrix $HM$ using $(x_t, y_t) = (-x_{k_t} \times (N/4) + N, z_{k_t} \times (N/4))$, with $x_k$ and $z_k$ the coordinates using the global reference. Starting with every position of the matrix $HM$ equals zero, every detection made at instance $t$ increments its value, as well as its 8 neighbours: $M(x_t + i, y_t + j) = M(x_{t-1} + i, y_{y-1} + j) + 1$ with $i$ and $j = \{-1, 0, 1\}$. After this step, the matrix $M$ holds values of the positions of a user, or a group of users (optional). The matrix is then normalized: having the highest value $h_m$ of matrix $HM$ with $h_m = \max(M(x, y))$, a grey scale image $H_g$ is calculated with $H_g(x, y) = (\tau / h_m) \times HM(x, y)$, with $\tau = 255$ in this case, visible on the bottom left row for the case of 1 user Fig. 4.3. Finally, the map can be converted to JET colour map, visible on the bottom right (same figure) in this case with several users (4) moving during 5 minutes.

**Sound**

Interaction with the installation through sound is still possible even if a user is not detected/present in front of it, the installation is capable of detecting the sound source angles and interacting with users via sound. As mentioned in Section 4.3.1, speech recognition can be done by a single Kinect, however in a 4 Kinect configuration ($180^o$) we have a total of 16 microphones ($4 \times 4$), so the Kinect that is most frontal to the sound source location must be selected. A single Kinect can determine the audio source direc-

tion from within $100^o$ range in front of itself. In a 4 Kinect configuration, where the sensors are positioned side by side and misaligned from each other, the total audio source direction range is calculated by adding $100^o$ from each angle range from each Kinect minus the 3 overlapped audio source direction ($Oa$) (see Fig. 4.1 bottom left) created by the close position of the Kinect from each other (see Section 4.3). In this 4 Kinect configuration the total audio source direction is $Sd \approx 223^o$ ($Sd = 4 \times 100^o - 3 \times Oa$), which is greater then the field of view of the combined Kinects RGB and depth data of $180^o$ degrees. So in order to detect sound from within the $180^o$ range, a transformation was applied to all audio beam angles, transforming into $180^o$, $\varphi_t = \varphi_o \times 180/Sd$, where $\varphi_t$ represents the transformed and $\varphi_o$ the original audio beam angles. This transformation will distorts the actual location from the sound being captured, however because we just want to select witch Kinect will be responsible for handling speech recognition this distortion will not influence this selection. A different possible solution would be to clip all information/sound that comes from audio source inside the range $[-(Sd-180)/2, 0]$ and $[180, 180+(Sd-180)/2]$ degrees.

When a user is interacting with the system using their voice or when environmental sound is present, to determine witch Kinect will handle speech recognition the follow algorithm is applied:

(a) Capture the sound beam and confidence levels from all 4 Kinect sensors.

(b) Sum and store the result of the individual audio beam angles multiplied by the respective confidence levels.

(c) In order to filter out isolate sound locations, step b) is repeated several times (it was used 10).

(d) In the end of step c) (all repetitions/sums) is determined which of the 4 Kinects has the greatest sound source locations times confidence level stored.

(e) In case of a tie or close tie (less than 5% of the highest value), the chosen value

will be the one that has the sound beam closest to $0^o$ (corresponding to the most frontal sound in respect to a Kinect position).

(f) Activate Kinect speech recognition from the selected Kinect correspondent to the maximum value calculated.

Fig. 4.3 top right shows real time sound source beams from all 4 Kinects, in a polar plot. In this case the Kinect selected is the right most one. As mentioned, the Kinect works by analysing statically stored keywords that can be detected with a degree of certainty every time a user speaks the keywords. When a keyword is identified and confirmed the application triggers a response using a stored answer (using an implemented text-to-speck functionality) that are replayed using a synthesized voice. The keywords can also trigger actions similar to the gesture interaction. If there is sound and no word is recognized in the followed 30 s, then the installation return a audio personalized message to call the attention to itself (this is only done if no user is interacting using gestures).

### 4.3.3 Database and Interface Module

The database module stores two types of information: (i) menus configurations and (ii) statistics. The Interface module is responsible for reading information stored in the database and automatically generate layouts, as well as generating responses to the user input. In resume, the application can generate the following layouts: (a) titles of menu contents useful to serve as bridge to other menus, (b) similar to a), a layout that can also fit small description if needed. In specific cases, (c) media content as image or video with description is useful to give a user the idea beyond a concept, generating a layout similar to b). Alternatively, the same layout can be used but displayed in e.g. diagonal instead of vertical. (d) One final option is available to display only images or video. All configurations are loaded when the application starts. The avatar or video trigger responses to inputs of the user: (a) If the interacting user with the in-

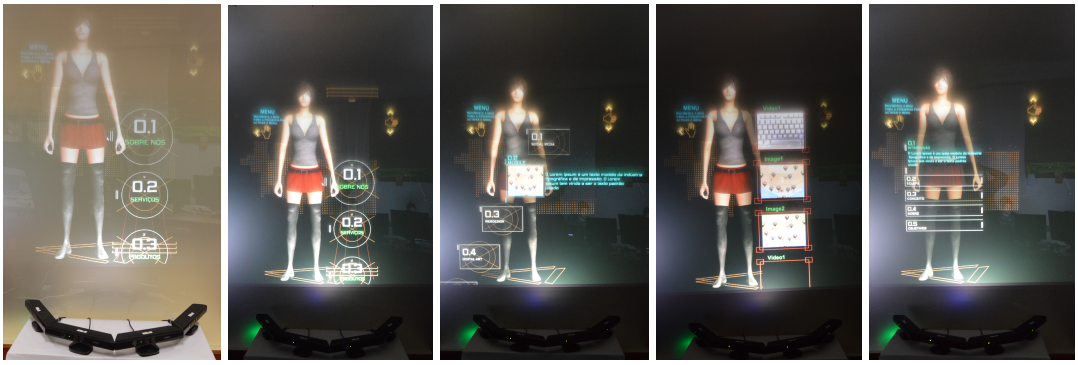

Figure 4.4: Several examples of the system interface (see text).

stallation changes, the avatar/video waves while showing that user's photo. (b) If the user selects an option, the avatar/video touches it's content triggering a menu change. All interaction, time spend interaction with the menus, request asked to the installation, etc. are saved in the database. Fig. 4.4 shows some examples of the prototype installation.

## 4.4 Discussion

In this Chapter an interactive installation was presented, based on a model for natural interaction. An avatar work's as a public relations giving the first contact between a company and a potential client, capable of recognizing sound sources, voice commands and interacting with sound and gestures. The fully customization of the application is a major asset, which can also extract statistics about users reached, positions at any instance of time, the most requested contents as well as time spent by users in each of the contents. The installation works in $180^o$ environment, recognizing up to 24 users, with a maximum of 8 users fully tracked. When a user is interacting with the installation and gestures' joints are tracked, since Kinect returns the three dimensional position of the joints relative to it, the distance of the joints to Kinect, as well as more users on the interaction zone, does not seem to interfere with the success rate at the specified distance and speed parameters (see Section 4.3.1). The results of swipe gestures were good, as previous hand positions were taken into account. The detected

successful rate was 98%, being too difficult to tell exactly what did fail. Usually it was due to the user not making the gesture correctly, due to a lack of speed or gesture extend (distance). Users could navigate easily through the menus, although some users were not familiar with the pose gesture. The results of recognition of multiple detections of a user showed also good results, even in complex environments (e.g. Fig. 4.1 bottom right). In about 45 minutes of continuous with multiples users entering and leaving the field-of-view of the installation two conclusions were taken: (a) If the user progressively changes his/her position and is facing the installation then no errors (0) occur, but (b) if the user either changes abruptly (e.g. run) his/her position or is not facing the centre of the installation, then the results drastically decrease. These errors are major due to the overlapping area of two Kinects being small, in case of an abrupt change of position, and due to the limitations of Kinect not detecting well users when they are sideways to it, situation that cannot be controlled. On the other hand, results on recovering users were satisfactory. If a Kinect is obstructed and a user does not move, then it works as expected, as long as the obstruction doesn't take to long, but if it is not the case, i.e., the user moves to a different location, the results are not reliable. Also, if another user switch place with the obstructed user, then he/she will be recognized as the user they switch places with.

Future work includes improving obstruction and overlap problems, using biometric information, such as distance between joints, and face recognition (when possible) to minimize the problem where two users switch places and, if the results are good enough, use the same landmarks to try and search for an obstructed user on a predicted area based on his past movements.

# 5

# Interactive 360° Holographic Installation

## 5.1 Abstract

With new marketing strategies and technologies, new demands arise, the standard public relation or salesperson is not enough, costumers tend to have higher standards while companies try to capture their attention, which is not always easy, requiring the use of creative contents and ideas. For this purpose, an interactive installation was developed, making use of a holographic technique to call the attention of a higher amount of potential clients, working as an host or showing a product advertising of the company (or even a face). The installation consists in a 360° (degrees) holographic avatar or objects and a screen, or group of screens around the installation, where a

set of menus with videos, images and textual contents are presented. It uses several Microsoft Kinect sensors for enabling natural interaction through gestures, sound and speech while building several statistics of the visualized content, number of people around the installation as well as heat maps. All those statistics can be analysed on-the-fly by the company to understand the success of the event.

## 5.2 Introduction

No matter what field a company works on, the most important of it's marketing strategies is costumer acquisition. Nowadays the first contact between a company and a client is mostly over their website, presenting on it a critical and precise information on what it has to offer, but often leaving unanswered questions or not capturing the client's total attention at all.

A public relation (PR), or a salesperson, however has more personal contact with the client, helping and establishing a more solid link with them while answering and persuading the client on what the company does and why it is better for that specific client. Exhibitions and other public events are often a great place to get new clients, with companies making their presence through a small group of PRs, relying then on creative ways for standing out from other present companies. However, nowadays with technological advances this seems not be enough usually forcing the company, to excel from others, on spending a lot of resources on disposable material combined with technological innovations.

For this last point, this Chapter presents a creative holographic installation that combines Pepper's Ghost holographic technique (Sprott, 2006) with natural interaction (NI). The installation consists in a new 360° (degrees) holographic representation of volumetric avatars or objects, and a screen or group of screens where a set of menus with videos, images and textual contents are presented. It uses several Microsoft Kinects (Kinect, 2014b) for enabling natural interaction through gestures,

sound and speech while building several statistics of the visualized content, number of people around the installation as well as heat maps. Those statistics can be analysed by the company to understand the success of the event. The hologram can work as an 360° host that follows user movements or shows the advertising of a product of the company or even a face.

The main contribution of this Chapter is the 360° holographic interactive installation, consisting in: (a) a new proposal for the 360° holographic representation, the (b) 360° interaction around the installation using 8 KinectS, which is highly customizable. (c) The navigation, achieved through gestures or voice commands by selecting the appropriate Kinect. (d) The user tracking, and a new user recognition as well as other persons around the installation (360°), and (e) building on-the-fly of several statistics of visualized contents and time spent looking to each content by various or a single user, including heat maps of the user and other persons standing around the installation.

In this section an introduction was made, in Section 5.3 the state of the art is presented. In Section 5.4 the structure of the installation ("hardware") is introduced , while in Section 5.5 is explained the developed work of the natural interaction, people tracking and recognition, statistics, and is explained briefly the data storage and the interface ("software"). In Section 5.6 an overview of the results and the prototype installation is presented. Finally, the conclusions and future work are presented in Section 5.7.

This work was done in conjunction with a Portuguese creative technological company: SPIC - Creative Solutions.

## 5.3 State of Art

Several applications or installations have been already developed through the years using holography (Antonio et al., 2013; Mihaylova, 2013). One of the most popular

techniques is Pepper's Ghost, due to it's simplicity of creating a hologram illusion just by reflecting an image using an acrylic or mylar foil placed by a 45° angle from that image, displayed on a LCD or on a surface via projector.

This simple technique was first used and created by John Henry Pepper, for performing magic tricks back in 1862. Figueiredo et al. (2014b) uses it for teaching 3D design, applying this technique for a better visualization of mechanics parts by students. On the other hand, the music business have been applying it to create concerts with dead artists, with the Tupac concert (Rennie, 2014) being one of the most famous.

Other applications were developed with similarities to the installation presented in this Chapter, when taking into account applications for commercial purposes. Flyway (2015a) makes use of Pepper's Ghost technique, being able to present a music concert with two holographic musicians. To create this illusion a projector is placed between the audience and the stage, projecting the characters to the stage using a mirror system. Virtual public relations is also featured in a holographic form, whereas AVA (Advanced Virtual Avatar) uses a real size PR in a holographic form (Flyway, 2015b). However, none of this solutions are completely 360°, capable of interacting with users through gestures and neither the hologram can react to user's position or voice.

Interaction with holograms also exists in applications such as D'Strict (2014) 3D Sensing Hologram Installation, which enables interaction through gestures with a hologram and a monitor placed inside a box, but is only visible in one perspective. Active8-3D (2015) features a medium or a large hologram only visible when placed in front of it, allowing very limited interaction. Another solution offered by them is a 3D-Holopyramid with the advantage of being visible in 360° with a total of 4 views due to its pyramidal shape. Other 360° hologram solutions exists and Lifefact Magic Displays (Litefast, 2013) uses a high speed rotating mechanical system to build such hologram, increasing the maintenance required. This installation doesn't offer interaction with the users. Vizzo also offers a solution on this matter with Cheoptics360 prod-

uct (Vizoo, 2015), also with 4 different views – pyramid, of an object being showed as a hologram using Pepper's Ghost, but it also offers no interaction.

Several other examples of installations with some similarities to the one presented in this Chapter exist, e.g. Paradigm (Rearpro, 2015) and more recently Holus (Holus, 2015) appeared, with the last one a prototype still in development by a Canadian start-up and under financing through Kickstarter consisting on a see-through tabletop box, presenting a 3D digital world users can interact with. Both of these solutions make use of either 1 of 4 perspectives and lack all the interaction features presented in this Chapter. Our previous work with an holographic installation can be seen in Alves et al. (2015b).

In the present Chapter a 360° holographic installation is introduced, different from all the previous, with 8 views / perspectives, instead of the usual pyramid representation with 4 views, allowing a better and more impactful visualization of the hologram.

Natural interaction (NI) uses several different types of sensors and cameras. The three-dimensional (3D) sensors are gaining more focus than all the others, due to their low prices and a great functionality when it comes to NI. Sensors such as Asus Xtion (Asus, 2014), Microsoft Kinect (Kinect, 2014b) or the Structure Sensor (Structure, 2014), enable hands-free controlling of devices or graphic user interfaces, tracking users position, recognizing skeleton's joints between other functionalities (e.g., recognition of voice commands). Others exist with Leap Motion (Motion, 2014) being also a 3D sensor and one of the more notorious, but is more focused on the recognition of user's hands or objects placed on it's sight.

Although all 3D sensor shares similarities, Microsoft Kinect is one of the most famous on the market and was popularized due to the video gaming industry. Lots of applications in different fields can be found using it: for robotics (El-laithy et al., 2012), for head pose classification (Yun et al., 2014), for art installations (Alves et al., 2014; Weiss Cohen et al., 2015), applied in assistive technologies such as enhancing visual performance skills on older adults (Chiang et al., 2012), or for operating wheelchairs

(Kondori et al., 2014). Our previous work shows rear-projection full size PR installation that allows 180° interaction using 4 Kinects (Alves et al., 2015a).

For the installation presented in this Chapter, 8 Microsoft Kinect sensors were used to enable interaction, user detection and tracking on 360° field of view around the holographic installation.

## 5.4 Installation Overview

As mentioned in the introduction, the installation was designed with the purpose of showing a 360° view of an hologram as well as enabling interaction and statistics (also in 360°) around the installation. For the development of the hologram, the Pepper's Ghost technique was used, since it is widely known and not very expensive, with also being easy to maintain (no mechanical parts) and / or transport and install it in any place. As mentioned in the state of the art (Section 5.3), the installation used is an improvement to the so call "pyramid" (4 views) Pepper's Ghost. This type of installation is very easy to project and mount, nevertheless the continuity of the hologram is not as advantageous with lesser perspectives.

More perspectives of the hologram, creates a better continuity in the hologram visualization, but decreases the size of the hologram, visible for the public, for the same dimension of the installation. Several tests were done with hologram views from 4 to 10. Taking advantage of the expertise in this area (digital marketing) of the commercial company (SPIC) that is co-working in the R&D of this installation (product), it was decided that the 8 views will have the most impact, limiting the discontinuities (see Section 5.6), and at the same time showing a good sized hologram.

Different from the pyramid installation, the proposed installation / structure is not trivial to implement, and to the best of our knowledge there is no scientific (or other type) publication showing how to dimension and implement this. The remaining of this section is dedicated to this purpose, to the inclusion of the Kinects sensors in the

structure and respective overlapping field of views - “Hardware Component”, Section 5.5 presents the “Interaction and Statistics Component” (software component).

As mentioned, the basic principle of the Pepper’s Ghost technique, consists in a flat surface retaining or emitting the image placed with an angle of 45° from a transparent foil sheet or thin acrylic glass (myler foil will improve the hologram visualization, but will increase the price) (D’Strict, 2014; Figueiredo et al., 2014a; Mihaylova, 2013; Moser, 2014; Sprott, 2006). Instead of just one acrylic glass, the 360° installation have 8 faces (views) of this material with a thickness of 2 millimetres, joined together around an upside axis, with them being atilt 45° from the flat surface, as the Pepper’s Ghost technique demands. To keep the inclination of the faces fixed, three different 3D (thin) parts were designed and 3D printed to create the supporting structure.

It was designed two different joints / junction models and a spacer printed black afterwards, visible in Fig. 5.1 on blue and yellow in order to observe their curves more easily. To keep all the junctions fixed, a long U shaped piece of aluminium was used on the top and bottom parts of each acrylic foil and used the L shaped piece of aluminium between the acrylic foils. Fig. 5.1 shows the 3D models and two pieces of aluminium used to keep the structure fixed. In the same figure bottom two rows shows the final model of the top structure (reflection surface), with it’s main goal being the creation of the 8 views of the hologram. The hologram is visible due to the reflections of the views of an object, which are created and displayed on a flat surface using a projector on it’s bottom (explained later). The main difference to the regular Pepper’s Ghost technique is a replication of 8 times around the structure, one for each face (each face have a different perspective).

It was decided to project the prototype installation with $d_1 = 1$ m (metre) of diameter on the top side and $h_o = 30$ cm (centimetres) of height (transparent hologram structure; Fig. 5.1 bottom). This distances were pre-determinate as good dimensions for projecting a visible hologram inside (for prove of concept) that can be seen easily until 5 m around the installation. All the calculations presented below can be extended

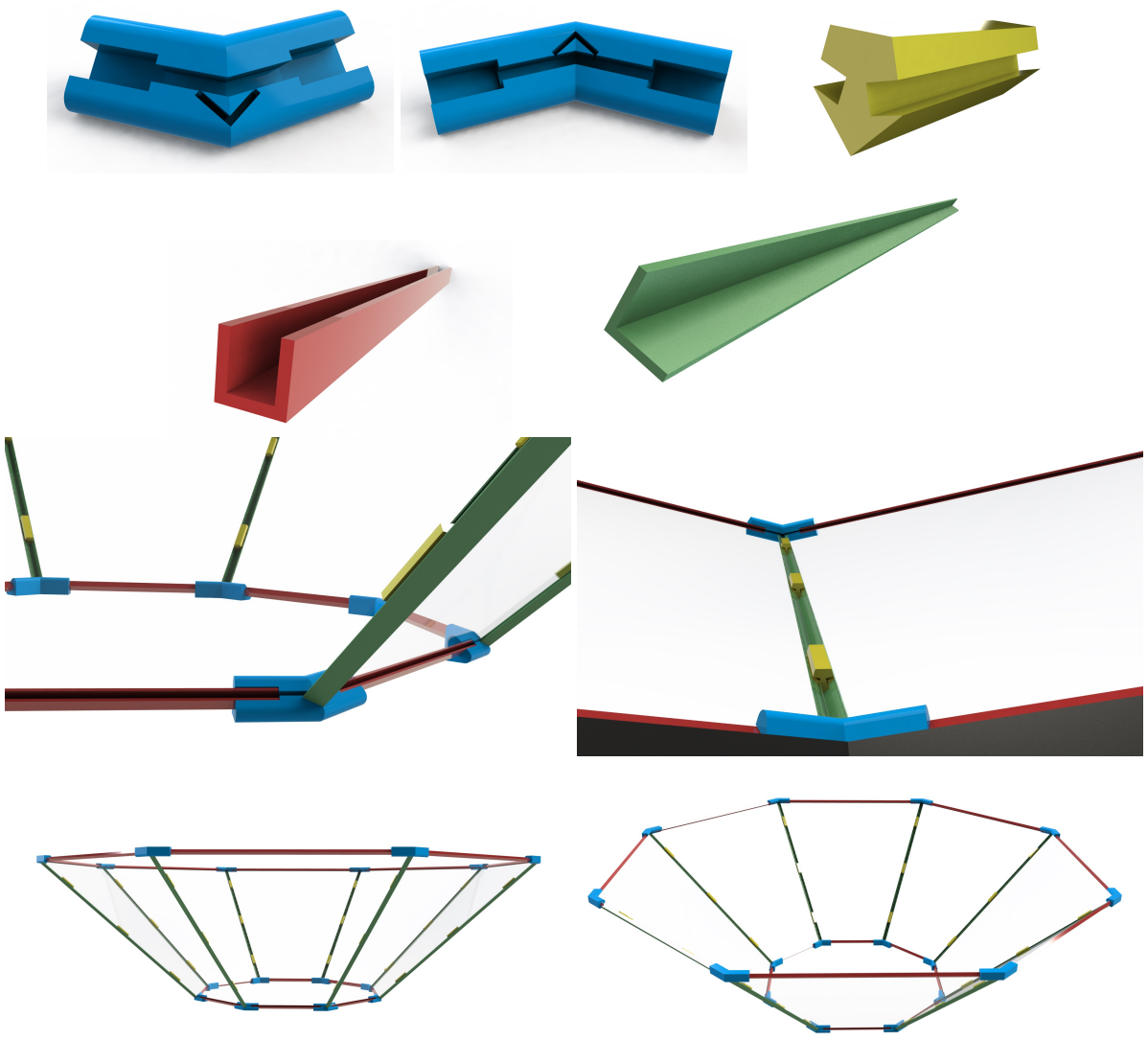

Figure 5.1: Three 3D models and two pieces of aluminium for keeping the structure fixed. From left to right, top to bottom, the bottom junction, top junction, spacer to keep all the faces spaced evenly (yellow), a U shaped piece of aluminium (red) and a another L shaped piece of aluminium (green) to join all the junctions and keep the faces fixed. Bottom two rows, 3D model mounting details of the reflection surface.

for any other dimensions.

With 8 faces, each top edge of the octagon shape have $l_1 = 41.4$ cm of length, $l_1 = d_1/(1+\sqrt{2})$. As each face is atilt 45°, the octagon on the down side was calculated with a diameter of $d_2 = 40$ cm, $d_2 = d_1 - 2 \times (h_o/\tan(45°))$. The length of the down side edge has $l_2 = 16.6$ cm, $l_2 = d_2/(1+\sqrt{2})$. Each acrylic face have a top and bottom edge length equal to the top and bottom octagon edge length, $l_1$ and $l_2$ cm respectively. As the face is atilt 45°, the height of each face is calculated with $h_f = \sqrt{h_o^2 + r_1^2}$ and $r_1 = (d_1 - d_2)/2$, with $r_1$ being the distance from the down side octagon to the edge of the projection surface (see Fig. 5.2 first row left).

Due to the weight and height of the installation and the projection area, a ultra short throw projector was chosen and placed below (in the bottom of) the structure, projecting upwards onto a flat surface that retains the image. The retaining surface ("acrylic glass rear projection screen") has the same dimensions of the top side of the structure, $d_1$, and the same octal shape.

As shown on Fig. 5.2, top left, the black bottom octal part is the rear projection retention surface. Each side of the structure reflects a part of the image in the retention surface, with the image being created by the single projector placed below, facing up and managed with the computer to divide the object (volume) in the 8 parts (views), one for each side. Doing this, an object appears to be floating inside the structure at a distance, behind the reflecting foil, equal from the reflecting foil to the retention surface. For this reason, to show the hologram in the middle of the structure, the retention surface needs to be distanced (height) $d_3 = 20$ cm ($d_3 = d_2/2$) from the acrylic foil; see Fig. 5.2 top left.

Creating an 360° hologram with 8 views / perspectives leads to perspective interferences. These interferences are due to adjacent sides and don't occur on the pyramid implementation, with a side reflecting part of the neighbours images (one perspective image), dropping the hologram quality if not dealt with. To correct this problem, placing a light blocking structure between perspectives is needed and done in two steps:

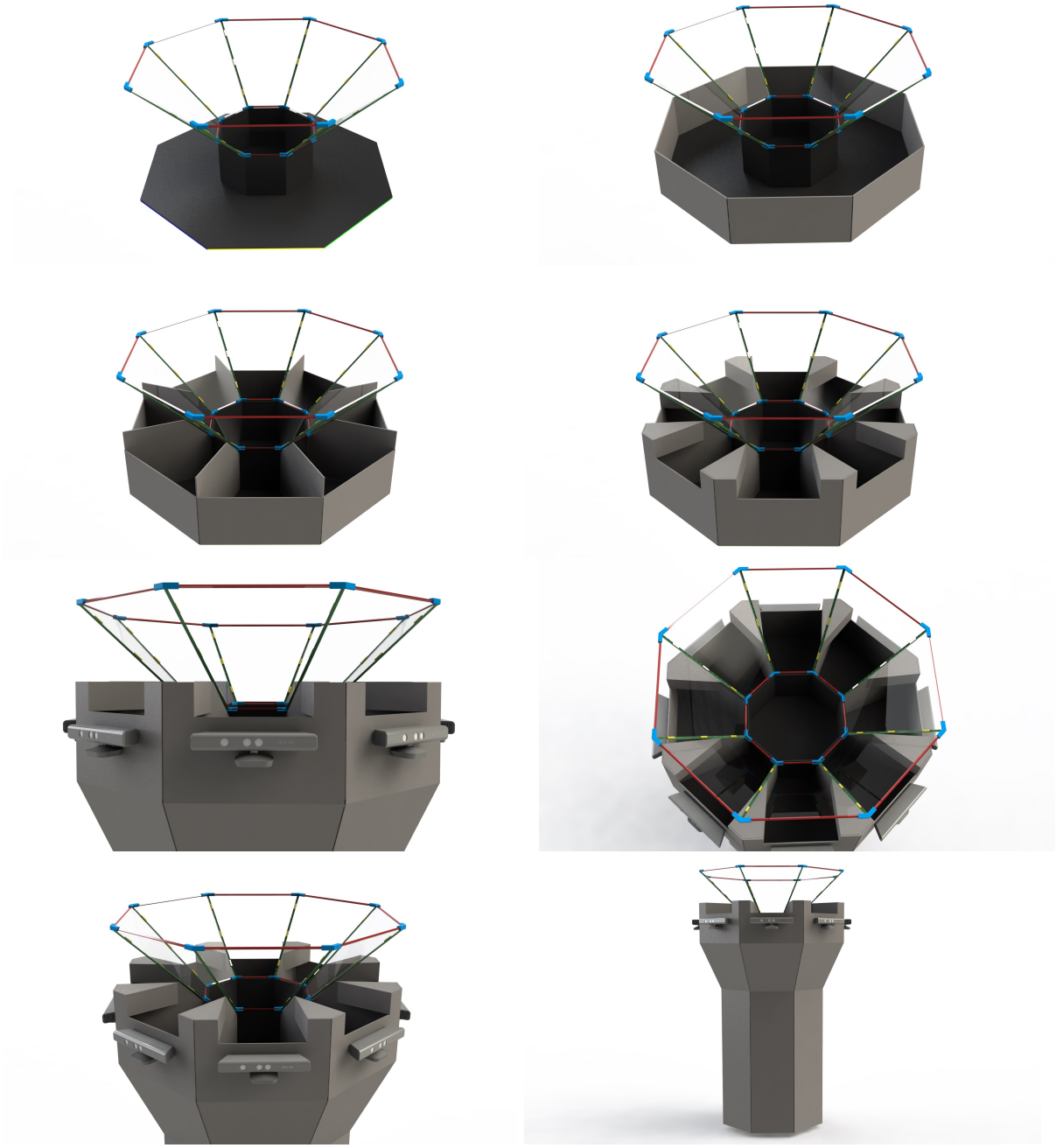

Figure 5.2: Placement of the rear projection display and construction parts for removing perspectives interferences. From left to right and top to bottom, screen placement (rear projection screen), blocking lateral light with structure of aluminium, division with blocking parts between perspective images, full designed dividing blocks and final top model with Kinects cameras positioned.

(a) placing a vertical piece of solid thin material above and between perspectives on the rear projection screen as shown on Fig. 5.2 top right and second row left, blocking most of the light between perspectives, the height is $d_3$. (b) Inserting on the top of the light barriers mentioned in (a), a "slice of cake" volume / shape, as seen on Fig. 5.2, second row right, enable blocking the remaining interference. The height and the inside angle of the isosceles triangle in the "slice of cake" shape was determined empirically after testing with several structures of different dimensions. The height is always less than $d_3/2$, in this structure the minimum value that can be used is 7 cm, and the angle is 35° with the bisectrix being $r1$. The remaining measures to create "slice of cake" volume are now easy to compute.

This corrections allows the user to look directly at the junctions between two acrylic faces, and see the continuity of the hologram both in the half left and right faces (not possible with the pyramid installation). With this, the structure eliminates completely any perspectives interferences and the user can move around the installation without any interference and with smooth transitions between perspectives of the hologram (see the practical results in Section 5.6).

The structure used as prototype has a height of 1.9 m, but can be adjusted in function of the company target, although this is the height suggested for an installation for medium size adults (from 1.65 to 1.85 m), where the hologram will be more or less in the line of sight of the viewer. However, for example, the structure should (of course) be lower if the company's target are kids. It is also important to state that the computer and projector are inside of the structure (below the holographic structure shown in Fig. 5.2, fourth row right), and the bottom shape of the structure can have any form.

Having now the holographic structure completed, the Microsoft Kinect sensors for the interaction were placed in the middle-front of each face. In Fig. 5.2, bottom row shows the position of each Kinect as also seen in Fig. 5.3 bottom row. In all illustration the 8 Kinects are shown in the top of the installation, but they can be placed in any height in the respective face as long as they're placed at the same height.

As mentioned, 8 Kinects were used, each having a field of view (FoV) of 57° (horizontal) × 47° (vertical) causing overlapping areas between each two adjacent Kinects by $\alpha = 12°$. This overlapping angle can be calculated by multiplying the number of Kinect by their horizontal FoV ($\lambda = 57°$), subtracting 360° and dividing by 8, $\alpha = (8 \times \lambda - 360°)/8$ (Fig. 5.3 top-right).

The bottom row of Fig. 5.3 shows the representation of the installation with the 8 FoV of the Kinects sensors, representing the existing overlap and gaps. Also shown is the position where the screen was placed to show the menus, etc. Once again, it is important to mention, that can be used from 1 to 8 screen, with any size. If desired, the screens can cover the entire bottom part of the installation, the interaction (see next section) will work with the user in any position in front of the installation, with all monitors showing the exact same output. Also, for the commercial installation the Kinects sensors can be hidden inside the structure of the installation.

## 5.5 Interaction and Statistics Component

The software component of the installation is divided in two main modules: (a) Users Data Module, responsible for handling and storing data received by Kinect sensors (main focus of this section), and the (b) Visual Output and Database module, responsible for generating all layouts and responses from the users input, storing data received by Users Data module and providing layout settings and information to the visual output.

### 5.5.1 Users Data Module

Users Data Module interacts directly with all Kinect sensors, receiving data from users' skeletons manipulating it for statistics generation, gestures or keywords recognition (sound) and even take users' photos. One of the biggest limitations of the Kinect is it's range, covering up to a distance of 4 meters with a maximum horizontal angle of

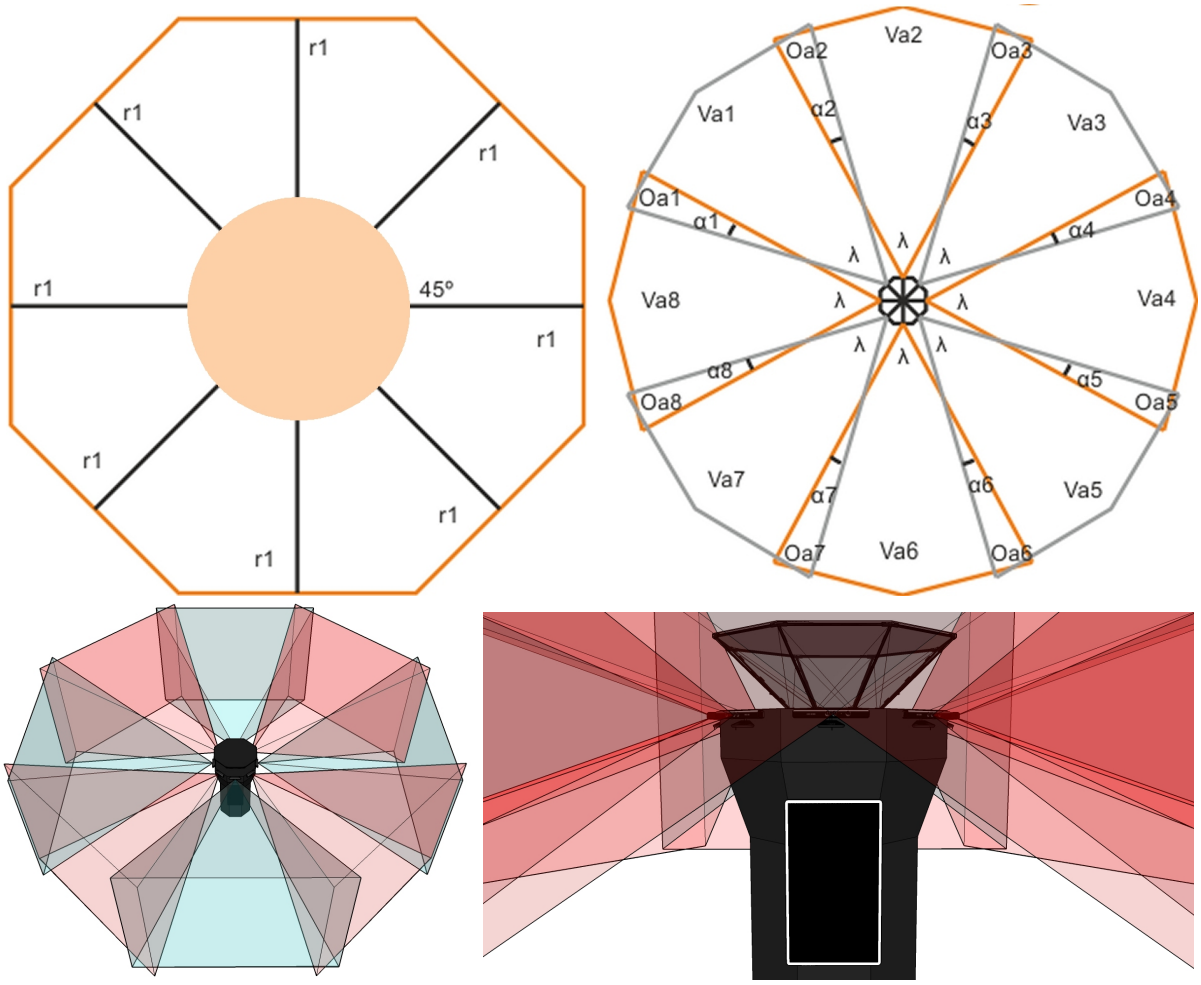


Figure 5.3: Top row, the physical coordinates and angles relative to each Kinect. Bottom two rows, the representation of the installation with the 8 FoV of the Kinects sensors.

57°. By combining 8 Kinect sensors, all area around the installation is available for interaction with users (ring centred in the middle of the installation from 1.1 m to 5.1 m; see also Fig. 5.8 left), also increasing the number of users' position from 6 to 48 and the number of users fully tracked from 2 to 16. These numbers, however, are the best case scenario since each Kinect is limited to 6 maximum of detected users' position and limited to 2 fully tracked users and, furthermore, there are intersection areas between adjacent Kinects (as shown in the previous section), causing two Kinects sometimes recognizing the same user. Before the recognition of a user by two adjacent Kinects, Users Data Module deals with each Kinect incoming data separately as shown in Fig. 5.4 - Single Management, being $i$ the sensor number ranging from 0 to 7. After this procedure for every sensor, all data retrieved by the single management block is translated to a global reference and users detected by two adjacent Kinect sensors are dealt with, Fig. 5.4 - Global Management.

Although single and global management use different algorithms, they share some similarities. All Single Management (SM) $i$ blocks, as well as Global Management (GM) block, have two First In First Out (FIFO) lists, one for the current active users being $Sc_i$ in case SM$i$ block or $Gc$ for the GM block, and one for the lost users being $Sl_i$ in case SM$i$ block or $Gl$ for the GM block. These lists store all information from Spatial Information block, explained in detail below.

**Spatial Information**

At approximately 30 times per second all Kinect sensors provide skeleton information about detected users, providing $(x,y,z)$ coordinates of detected users' joints and local position, i.e., a Kinect returns a coordinate relative to it, with the $x$ component being the horizontal distance to it, the $y$ component being the vertical distance to it and the $z$ component the depth distance to it. Every time a Kinect sensor $i$ has new skeleton information available, three different situation may occur (on the Kinect local $Sc_i$ and $Sl_i$ lists):

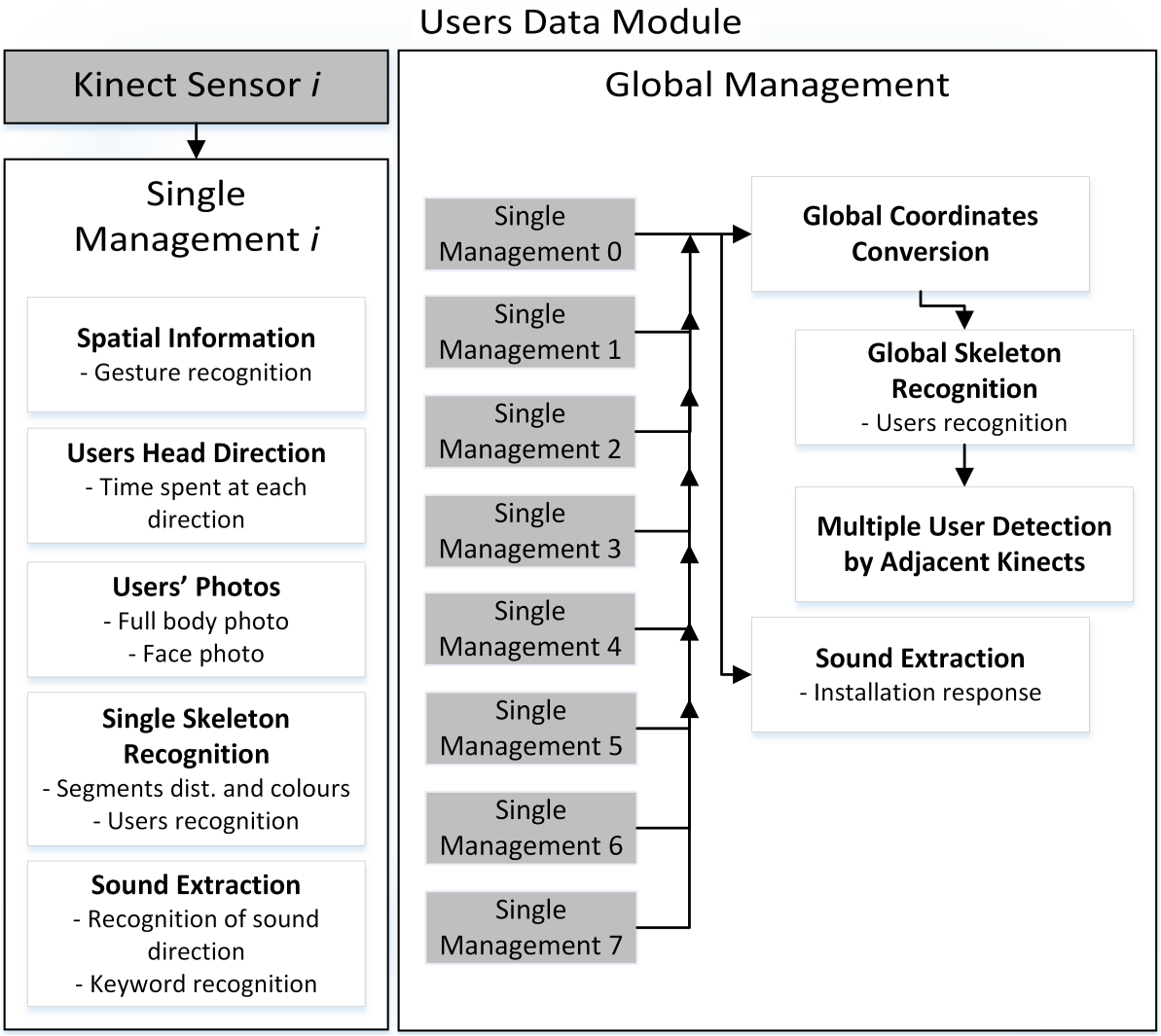


Figure 5.4: A detailed flowchart diagram of Users Data Module.

- ***New user detected***, meaning no information about this user exists on $Sc_i$. A new user is created assigning him an internal ID, which is then incremented, and all joints information and local position are added to this user alongside an entry time on $Sc_i$.

- ***User already exists***, with the current information belonging to an existent user on $Sc_i$. In this case, the information is added to this user alongside it's entry time.

- ***No new information on a existent user*** of $Sc_i$, meaning this user no longer exists. At this moment most of this user information is added or updated to the database (see below, Section 5.5.2), with the user being moved to the lost users list $Sl_i$.

Gestures recognition are an important part of a hands free system, allowing users to control the application interface. After experimenting with several gestures, the "menu pose" and "swipe gesture" were chosen due to intuitiveness, both visible on Fig. 5.5 left, respectively left and right in the illustration. The implementation of the gestures make use of the spatial information obtained over time by the above procedure, and are implemented with:

- ***The pose gesture*** is detected by using a vector defined by the user's body $\overrightarrow{V_{bd}} = (x_{bd}, y_{bd}, z_{bd})$, computed by subtracting the *shoulder centre* joint (Kinect, 2014b) from the *spine* joint, $J_{sc}$ and $J_s$ respectively, and using another vector defined by the user's arm, $\overrightarrow{V_{ar}} = (x_{ar}, y_{ar}, z_{ar})$, calculated by subtracting the *hand (right / left)* joint, $J_{hj}$, from the *elbow (right / left)* position, $J_e$. The user is performing the gesture if: (1) as the user must be facing Kinect sensor, the horizontal angle to the camera perspective $\varphi$ must be defined between -$10^o \leq \varphi \leq 10^o$, with $\varphi = \text{acos}((\overrightarrow{V_{ar}} \cdot \overrightarrow{V_x}) / (||\overrightarrow{V_{ar}}|| \, ||\overrightarrow{V_x}||))$ and $V_x = (1,0,0)$. (2) If the angle between $\overrightarrow{V_{ar}}$ and $\overrightarrow{V_{bd}}$ defined by $\theta$ is a value between $20^o \leq \theta \leq 160^o$, with $\theta = \text{acos}((\overrightarrow{V_{bd}} \cdot \overrightarrow{V_{ar}}) / (||\overrightarrow{V_{bd}}|| \, ||\overrightarrow{V_{ar}}||))$. (3) To increase reliability, a verification is made and checked if at least 85% out of the detections made in the past 1 second are

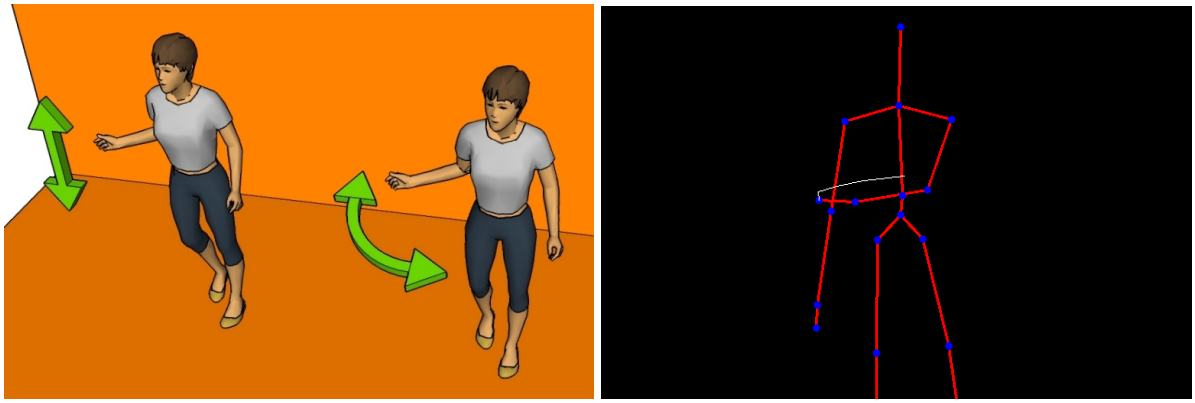

Figure 5.5: On the left, implemented gestures with the first performing menu pose gesture and the second the swipe gesture. On the right, the skeleton representation of a real person doing a swipe gesture.

according to (1) and (2). If true, the user is performing the gesture, with it being considered - up - if $\theta$ is less than $90^o$ and - down - otherwise.

- ***The swipe gesture*** is implemented using a minimum distance and velocity, $d_s$ and $v_s$ respectively. A time window $\Delta t_s$ can then be calculated with $\Delta t_s = d_s/v_s$, giving a value of past seconds that need to be analysed. By analysing user information acquired in the interval from $t$ to $t - \Delta t_s$, a swipe was performed if in any other subinterval $\Delta t_k$, between $t$ and $t - \Delta t_s$ and with $k$ the latest instance of $\Delta t_k$, all hand joints ($hj$) minus it's previous hand position, only taking into account the $x$ component for the horizontal swipe, and the $y$ component for the vertical swipe, has the same signal along the whole interval $\Delta t_k$, $(|\sum_{j=k-\Delta t_k}^{k-1} \text{sgn}(hj_{j+1,\{x/y\}} - hj_{j,\{x/y\}})| = k - 1)$, and if the total distance travelled in that subinterval, taking only into account the $x/y$ component respectively for the horizontal and vertical swipe, between the first and the last point of that sequence is greater or equals to $d_s$, $d_s \geq |hj_{k,\{x/y\}} - hj_{k-\Delta t_k,\{x/y\}}|$. Fig. 5.5 right shows a user performing this gesture. The minimum speed used was $v_s = 200$ cm/s while the minimum distance was $d_s = 30$ cm (this can be configuration if pretended in the database of the installation).

**Users Head Direction**

Detected users passing in front of the installation are not always paying attention at it. Users head direction estimation is useful for statistics, i.e., to know the time spent by a user truly interacting and paying attention to the installation, both the contexts (menus, images, movies, etc.) and hologram.

Microsoft Kinect sensor's low resolution of 640×480 px (pixels) is not reliable enough for capturing eyes direction at a distance between 1 to 4 meters to the sensor (Martins et al., 2015). Due to this and instead of doing that, an estimation of the user's head direction is done, with it being a good indicator of where the users' are looking to, in this particular application. To do this procedure, the 2D position of both *eyes* and *shoulder* left, right and centre are used, Fig. 5.6 top, with shoulders in black and eyes in red. First, the shoulders vector $\overrightarrow{V_s}$ is calculated, defining a line containing both shoulder left and right defined by $S_{\{l/r\}} = (x_{\{S_l/S_r\}}, y_{\{S_l/S_r\}})$. The vector $\overrightarrow{V_s}$ can then be calculated with $\overrightarrow{V_s} = (x_{S_l} - x_{S_r}, y_{S_l} - y_{S_r})$ and a normal vector $\overrightarrow{V_s'}$ can easily be found with $\overrightarrow{V_s'} = (-V_{S_y}, V_{S_x})$. The normal vector $\overrightarrow{V_s'}$ is then applied to the *shoulder centre* position defining an almost vertical line, visible in Fig. 5.6 top in blue. With this, one out of three situations can happen:

- Both eyes on each side of the line, meaning that the user is looking centre, visible in Fig. 5.6 top-centre.

- Both eyes on the left side of the line, with the user looking to the right, visible in Fig. 5.6 top-left.

- Both eyes on the right side of the line, thus the user is looking to the left, visible in Fig. 5.6 top-right.

Depending on the current position of the user, extreme left and right were excluded. A timer was then implemented to count the time spent looking at each direction.

Figure 5.6: Top, users head estimation. On the left, user looking to the right, middle, looking to the centre and on the right, user looking to the left. Bottom, users photo extraction. Left to right, a face photo of a user, a boolean mask containing only the user, a full body photo of a user, and a full body photo with no background of a user.

**Users' Photo**

Users interacting with the installation can take a full body or face photo of themselves (selfie) for "promotional gifts", or to appear on the users photos gallery or to send it to his/her mail and/or social network. In case of the face photo, Kinect SDK (Kinect, 2014b) does most of the job by providing a method that retrieves both upper and lower corner of the rectangle containing the user's face, Fig. 5.6 bottom first column. Full body photo, however, is not done directly, although there are several ways of doing so using a combination of built-in methods. Every time Kinect signalizes a new depth frame $Dp(x,y)$ is available, the first 3 bits, ranging from 0 to 6, indicates whether a pixel belongs to the background, having the value 0, or the pixel belongs to a user, having any of the remaining values. A boolean mask $B_u$, Fig. 5.6 bottom second column, can be built for every user $u$, with $B_u(x,y) = 1$, if $Dp(x,y) \bmod 8 = u$; 0 otherwise. For the image $B_u$ highest and lowest value $x_{\{min/max\}}$ and $y_{\{min/max\}}$ are found. The

full body photo $B(x,y)$, Fig. 5.6 bottom third column, can be easily taken by cropping the Kinect RGB frame $I(x,y)$ with $(x_{min},y_{min})$ and $(x_{max},y_{max})$. Optionally, a Gaussian filter ($\sigma = 2$) is applied to $B_u$, returning $Bg_u$, performing then a bitwise AND with $I(x,y)$ and cropped with the above procedure, returning the same full body photo but with no background, visible in Fig. 5.6 bottom fourth column.

**Sound Extraction**

Each Kinect is equipped with a built in microphone array capable of sound source localization, formally entitled beam forming. For each Kinect the audio beam angle range, covers 100° in front of itself, 50° from the centre to each side (Kinect, 2014b). The beam angle can be one of 11 different values, an integer multiple of 10, with each of this values representing sound source directions. In addition, the Kinect SDK also returns a confidence level of that estimation from a value between 1 and 0, were 0 represents no confidence and 1 maximum confidence. The Kinect SDK also supports speech recognition in several languages, allowing the detection of words or sentences. In the Global Management block, the sound localization and confidence returned by each individual Kinect is combined and is selected the Kinect that will be use for the speech recognition, see Section 5.5.1.

**Single Skeleton Recognition**

Once users are added to $Sc_i$, they start being tracked for recognition with past users. This is very useful for companies to know if a past user has returned to check the installation again and asked for more information about their products. This procedure consists on analysing the colour of users' skeleton segments and distances, creating a colour-biometric descriptor to recognize users that have been previously interacting with the installation. Obviously, if users change their clothes and appear some time later near the installation, they will not be recognized as a previous user. Although this is true, it is not a gross error since this statistics are desired mostly for a single day

session, being very unlikely a user doing this.

Kinect sensor provides us with 20 joints $J$, meaning a total of 19 segments are available, visible in Fig. 5.7 first row left (joints in blue, segments in red), although more segments could be used by combining joints that are not adjacent to each other. Every segment is defined by a vector $\overrightarrow{Vs}_h$, with $h$ ranging from 0 to 18, found by subtracting the two adjacent joints using $\overrightarrow{Vs}_h = J_h - J_{h+1}$. Note that the use of $(h+1)$ in the previous formula is an abstraction, since, in the practical implementation, using the Kinect SDK, the joints are not numbered in order.

Using this vector, is then calculated each segment region colour patch for the user/person descriptor, with the width of $wr = 0.02$ m, this value could be dynamic in function of the segments length, however tests showed that results do not improve by doing this. It is then, analysed always the same width even if the user is closer or further to the installation, with this region having a higher amount of pixels if the analysed user is closer to the camera than further from it, Fig. 5.7 second row.

To define this region (patch), a normalized normal vector $\overrightarrow{Vs}'_h$ must be found, with it being normal to both $\overrightarrow{Vs}_h$ and camera perspective $\overrightarrow{V}_z = (0,0,1)$. Vector $\overrightarrow{Vs}'_h$ is found with $\overrightarrow{Vs}'_h = \overrightarrow{Vs}_h \times \overrightarrow{V}_z / ||\overrightarrow{Vs}_h \times \overrightarrow{V}_z||$. The four corners of the region (patch) then found with $P_{\{1,\ldots,4\}} = \{J_h + wr/2 \times \overrightarrow{Vs}'_h, J_h - wr/2 \times \overrightarrow{Vs}'_h, J_{h+1} + wr/2 \times \overrightarrow{Vs}'_h, J_{h+1} - wr/2 \times \overrightarrow{Vs}'_h\}$. After converting $I(x,y)$ (see Section 5.5.1) to HSV ($I_{\text{HSV}}$), a histogram ($H$) is calculated for every segment region using all four corners above, $P_{\{1,2,3,4\}}$, as well as a normalised average and standard deviation of the distances of every skeleton segments (see below).

Users can sometimes be moving fast or interacting with the installation, Fig. 5.7 first row right, and the segment might not actually belong to the user but to the background instead, turning the distances and colours unreliable. To avoid this problem, segments are only marked for analysis if both joints $J_h$ and $J_{h+1}$ are not moving fast. This is done by checking if in the 5 past samples both joints are approximately stationary according to $|J_{\{h/h+1\}}(t) - J_{\{h/h+1\}}(t-1)| < 0.15$ m, with $t$ ranging from 0 to 4. If this

condition is true, then this sample is counted as successful and will be used to calculate both the $N_p$ histogram average $\overline{H}_{h,c}$ (over time), i.e., $\overline{H}_{h,c} = 1/N_p \sum_{k=q-N_p}^{q} H_{h_k,c}$, with $c = \{\mathrm{H}, \mathrm{S}\}$ being the colour channel of the histograms, $N_p = 60$ samples and $q$ the latest sample, and segment average distance and standard deviation. $N_p$ value was considered 60, this means that the algorithm needs 2 seconds at 30 frames per second to analyse a user. However, this value is increased to about 6 seconds due to the requirement of fully sampled segments ($T_{seg}$=16) needed while is also required for them to be approximately stationary during the sampling.

After this step is calculated the normalised average as well as the standard deviation of all vectors $\overrightarrow{Vs}_h$. First is calculated the average $\overline{D}_h = 1/N_p \sum_{k=q-N_p}^{q} ||\overrightarrow{Vs}_{h_k}||$. After this is found the highest value $D_{max}$ from all $\overline{D}_h$ with $D_{max} = \max(\overline{D}_h)$. The final step is normalizing all $\overline{D}_h$ by dividing them with the value $D_{max}$ giving $\overline{Ds}_h = \overline{D}_h / D_{max}$, and the standard deviation $\sigma s_h = \sqrt{\sum_{k=q-N_p}^{q} (||\overrightarrow{Vs}_{h_k}|| / D_{max} - \overline{Ds}_h)^2 / N_p}$.

With all segments analysed, the next step is to compare a current user/person with previous users/persons and check if there's a match between them, being "1" and "2" the "person1" and "person2". Before starting, is checked if at least $T_{seg}$ (16) segments analysed $N_p$ times are available. Only then is compared a 1st person histogram $\overline{H1}_{h,c}$, $\overline{Ds1}_h$ and $\sigma s1_h$ with 2nd person $\overline{H2}_{h,c}$, $\overline{Ds2}_h$ and $\sigma s2_h$. If this is not true, then the users are not compared until this information is available, but if they are is checked the following two conditions:

- ***Criterion (a)***: An average comparison of the available segments between each other is performed, comparing both H and S channels of the HSV histogram: $Ah_c = 1/At \sum_{k=0}^{At} \rho(\overline{H1}_{k,c}, \overline{H2}_{k,c})$, with $At$ being the total available segments and $\rho$ being the correlation between the two histograms for each channel. If $((Ah_{\mathrm{H}} + Ah_{\mathrm{S}})/2 \geq TP \wedge Ah_{\mathrm{H}} \geq TH \wedge Ah_{\mathrm{S}} \geq TS)$ then these two persons have similar outfit. Being $TP = 75\%$ and $TH = TS = 60\%$.

- ***Criterion (b)***: For each person, intervals defining the typical distances of the seg-

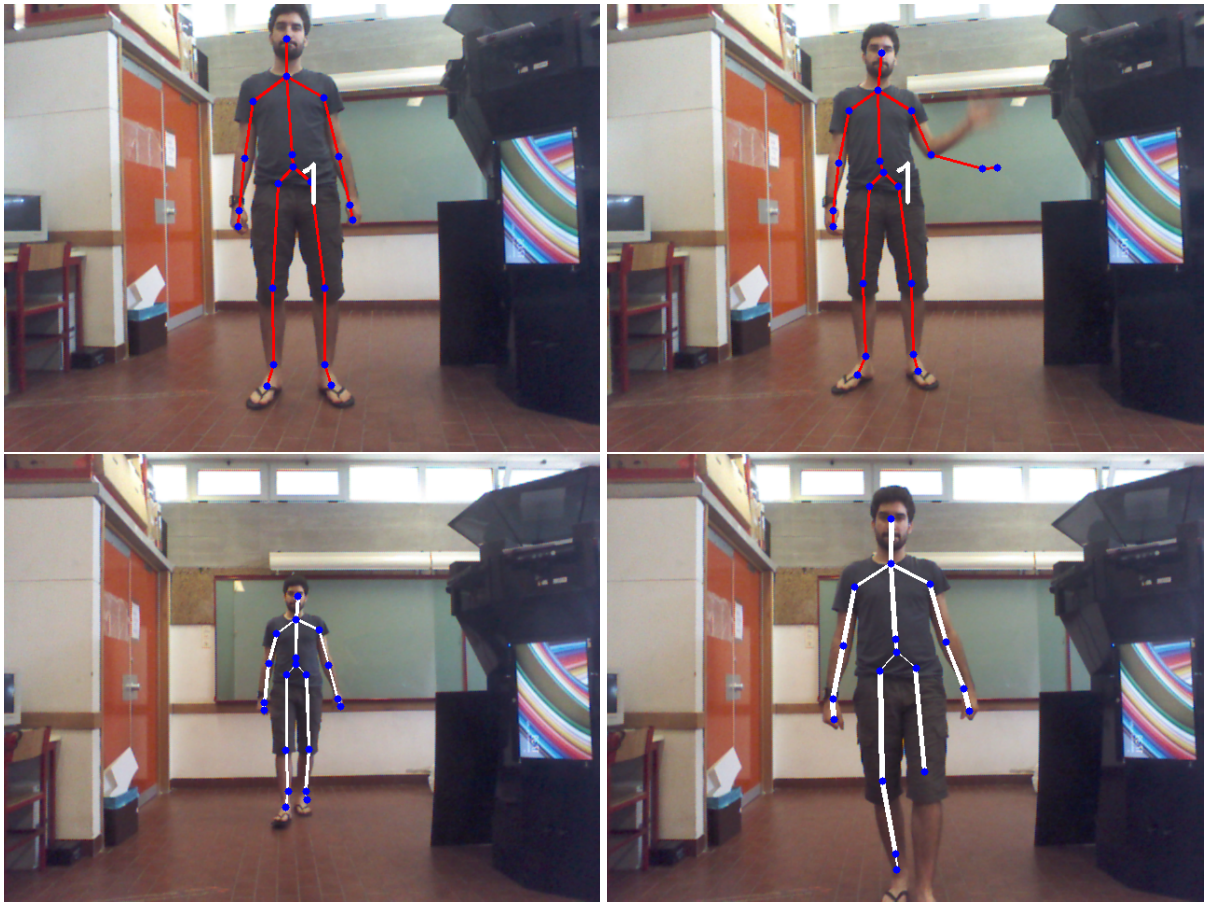

Figure 5.7: First Row, analysed skeleton segments, on the right, visualisation of a small delay problem. Bottom row, visualisation of the analysed area for every skeleton segments, with the pixel area increasing as distance decreases.

ments are computed, finding a $D\{1/2\}min_h$ and a $D\{1/2\}max_h$. The minimum values are found using $D\{1/2\}min_h = \overline{Ds\{1/2\}}_h - \sigma s\{1/2\}_h$ and $D\{1/2\}max_h = \overline{Ds\{1/2\}}_h + \sigma s\{1/2\}_h$. A counter $CI$ starting with zero was implemented and is incremented if the intervals intersects as $[D1min_h, D1max_h] \cap [D2min_h, D2max_h]$. If $CI/At \geq TC$ then is considered that these two users have similar skeletons, with $TC = 75\%$.

If the above two conditions are true then these two persons/users are considered to be the same. Despite the $TP$, $TH$, $TS$, $TC$ and $T_{seg}$ being calculated empirically, tests done in different environments showed that small changes do not affect the results, nevertheless, using higher values enhances the precision of the algorithm. Tests described in Section 5.6 shows no need for that.

**Global Coordinates Conversion**

Once all data retrieved by each Kinect are available and processed separately by the above procedures, the next step is to add, update or remove users from $Gc$ and $Gl$ lists. Before doing this, all users from $Sc_i$ need to be converted to a global reference and put on a potential users array $Uc$. Given the Kinect sensors physical placement $C_i(x,y,z)$, horizontal rotation $\beta_i$ and vertical rotation $\phi_i$ (with $i = \{0,...,7\}$), a Kinect coordinate can be translated to a global reference.

The vertical rotation $\phi_i$ is not needed to calculate, since Kinect is equipped with sensors that returns this angle. On the other hand the horizontal rotation $\beta_i$ must be calculated, as visible in Fig. 5.3 top left, $\beta_i = 45^\circ \times i$. With $\beta_i$ determined and given the radius $R1 = d_1/2 + K_w$ cm, with the $K_w$ being the offset of the Kinect in relation to the structure (for the present structure $K_w = 10$ cm and $R1 = 60$ cm; see also Section 5.4), the physical coordinates $C_i$ can be calculated using polar to 3D coordinates conversion with $C_i(x,y,z) = (R1 \times \cos(\beta_i), 0, R1 \times \sin(\beta_i))$, with the $y$ coordinate being 0 due to the Kinect sensors being at the same height and being used the same coordinate system as Kinect. For the present structure the final physical coor-

dinates are: $C_i(x,y,z) = \{(60,0,0), (42.4,0,42.4), (0,0,60), (-42.4,0,42.4), (-60,0,0), (-42.4,0,-42.4), (0,0,-60), (42.4,0,-42.4)\}$.

To convert the coordinates from single to global reference, are first found the 3 unitary directive vectors, representing each axis: $\overrightarrow{V_a}$, vector of the axis a Kinect is facing, $\overrightarrow{V_a}(x,y,z) = (\cos(\beta_i) \times \sin(\frac{\pi}{2} + \phi_i), \cos(\frac{\pi}{2} + \phi_i), \sin(\beta_i) \times \sin(\frac{\pi}{2} + \phi_i))$. $\overrightarrow{V_b}$, representing horizontal axis and making 90° to $\overrightarrow{V_a}$, $\overrightarrow{V_b}(x,y,z) = (\cos(\beta_i + \frac{\pi}{2}) \times \sin(\frac{\pi}{2} + \phi_i), \cos(\frac{\pi}{2} + \phi_i), \sin(\beta_i + \frac{\pi}{2}) \times \sin(\frac{\pi}{2} + \phi_i))$ and finally $\overrightarrow{V_c}$, representing the third axis with 90° degrees to both $\overrightarrow{V_a}$ and $\overrightarrow{V_b}$ pointing upwards in relation to Kinect, $\overrightarrow{V_c}(x,y,z) = (\cos(\beta_i) \times \sin(\phi_i), \cos(\phi_i), \sin(\beta_i) \times \sin(\phi_i))$.

Since Kinect SDK joint's coordinates $J$ are in meters while $C_i$ is given in centimetres, an auxiliary joint $\hat{J}$ is calculated with $\hat{J}_{x,y,z} = J_{x,y,z} \times 100$. The new position in the global reference $Jg$ can be computed with $Jg(x,y,z) = (\hat{J}_x \times \overrightarrow{V_b} + \hat{J}_y \times \overrightarrow{V_c} + \hat{J}_z \times \overrightarrow{V_a} + C_i)/100$ m, with $Jg$ being the new joint and $C_i$ the physical position of Kinect $i$. Any joint or position can be converted with this method to the global reference, and wherever this procedure is used a denotion of either $Jg$ for joint or $Pg$ for position is used. An aerial scaled representation can be seen in Fig. 5.8 left, with the grey areas being Kinects field of view, the lighter areas being intersection areas between two adjacent Kinect sensors and black areas the gaps. All converted users are added to $Uc$ list for verification of multiple detections by adjacent Kinects, explained below. Four persons are also represented in the same figure by colour circles, each with the diameter of 50 cm.

**Multiple User Detection by Adjacent Kinects**

As explained above, adjacent Kinects intercept partially their field of view (Fig. 5.3 and 5.8 left), meaning a user can be detected by two different Kinects. To solve this problem, all potential users on $Uc$ are compared with each other with two different criteria:

- If the euclidean distance between the global position of two users detected by different Kinects, $Pg_1$ and $Pg_2$, on $Uc$ is less than 50 cm (considered the "mini-

mum weight of a person") and if the last $T$ skeletons information available, $T = 5$ are similar to each other given by $(|Pg_{\{1/2\},x}(\delta) - Pg_{\{1/2\},x}(\delta - 1)| < 15 \text{ cm}) \wedge (|Pg_{\{1/2\},y}(\delta) - Pg_{\{1/2\},y}(\delta - 1)| < 15 \text{ cm}) \wedge (|Pg_{\{1/2\},z}(\delta) - Pg_{\{1/2\},z}(\delta - 1)| < 15 \text{ cm})$, with $\delta = \{t, ..., t - T\}$, then is considered to be the same user, assigning the internal ID of the user with the older entry time to the user with the newest entry time, also forcing this change on the list managed by $Sc_k$, with $k$ the Kinect detecting the newest user.

- If two users on $Uc$ already have the same internal ID, then the users are considered to be the same, as the relation was already found by the above procedure on a previous run.

In case two persons/users were detected as the same user, the one furthest to it's Kinect is discarded from $Uc$. All other users are marked for addition on $Uc$, can either be new users or information about previous users.

**Adding, Updating and Removing Global Users**

All users marked for addition $u_a$, $a$ ranging from 0 to the size of $Uc$, are compared with each of users $gc_e$ on $Gc$, with $e$ ranging from 0 to size of $Gc$. If both users $u_a$ and $gc_e$ share the same internal ID, then $gc_e$ will be updated with the newest information of $u_a$. In case $u_a$ does not have any internal ID correspondence with any $gc_e$ then $u_a$ is completely new to the list $Gc$ and is added to it. If, on the other hand, a user $gc_e$ does not have any correspondence with $u_a$ then $gc_e$ has been lost and is moved from $Gc$ to $Gl$.

Kinect's field of view can sometimes be blocked by a person passing by. A simple occlusion detection was built to recover a lost user on $Gl$. This procedure could use the algorithm explained in Section 5.5.1, however it was chosen not to, since in the early testing of the whole application it was found out that in most cases, Kinect is obstructed less time (typically 1 to 2 seconds) than the necessary to be completely

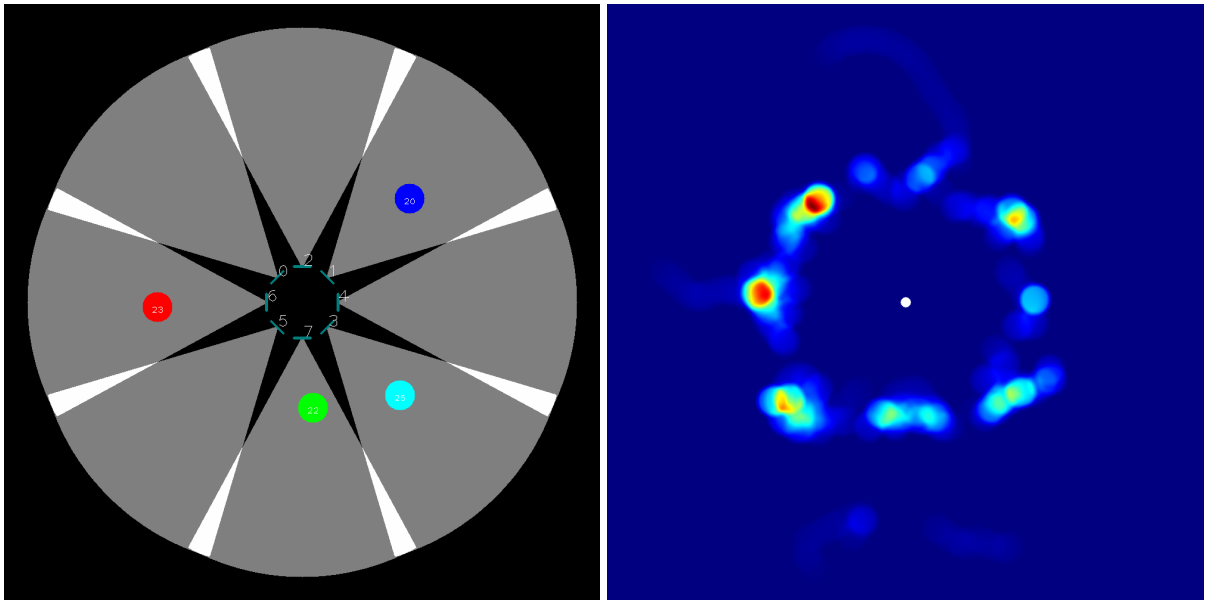

Figure 5.8: Global positioning aerial view on the left, and heat map on the right.

analysed by the mentioned algorithm.

When a new user is detected, instead of being added to $Gc$, a verification is made with users lost less than $\Delta t$ seconds ago. For all users lost in the past $\Delta t$ seconds, $gl_r$, and the new user $gc_q$ are compared and checked if the Euclidean distance $d_{r,q}$ between their global positioning is less than 50 cm. If more than one lost user $gl_r$ is according to this, is chosen the one with the lowest $d$, recovering all information to the current users list $Gc$ at the position corresponding to the user first entry time (it was used $\Delta t = 5$ s). The user interacting with the installation is the first user on $Gc$ corresponding to the user with the lowest entry time, remaining so until leaving the interaction zone or being idle during more than 30 seconds, in which case the entry time is changed to the current instance, reorganizing $Gc$ list.

### Global Skeleton Recognition

As previously explained on Section 5.5.1, every single management $i$ analyses and creates a colour-biometric descriptor of the people arround the installation or users. The methods used on that section are for a local Kinect (only). Kinects were not calibrated, their colour perception are different from each other, compromising the usage

of the same algorithm when comparing users that were detected by different Kinects. Due to this and since colour-biometric descriptors are already available, was complemented the previous algorithm that enables the recognition of users detected by different Kinects by comparing these descriptors.

Similar to the method used on Section 5.5.1, when comparing a current person1 of $Gc$ with a previously detected person2 on $Gl$, is checked if at least $T_{seg}$ segments analysed $N_p$ times are available (from both person1 and person2). If this is not true, the comparison is delayed until sufficient segments are available, otherwise their data are compared according to the following two conditions:

- First is performed $R_{h,c} = \text{IFFT}((\text{FFT}(\overline{H1}_{h,c})^* \times \text{FFT}(\overline{H2}_{h,c}))/(||\overline{H1}_{h,c}||\,||\overline{H2}_{h,c}||)$. With (I)FFT the (Inverse)Fast Fourier Transform, and $^*$ the conjugate. The result $R_{h,c}$ will hold the correlation value between histograms $\overline{H1}_{h,c}$ and $\overline{H2}_{h,c}$ while performing a shift operation on $\overline{H2}_{h,c}$, having the correlation value for every shifting performed. If the compared persons are the same, is expected that all $R_{h,c}(\chi)$ have their maximum values at approximately the same $\chi$ value. For this, is performed an average of the maximum values of all available $R_{h,c}$. The average of component S is computed with $\overline{S1} = 1/At \sum_{k=0}^{At} \max(R_{k,\text{S}}(\chi))$. The average of component H, however, cannot be performed of the same way, since H value is circular, the average was found with $\overline{H1} = \text{atan}(\sum_{k=0}^{At} \sin(\max(R_{k,\text{H}}(\chi)))$ / $\sum_{k=0}^{At} \cos(\max(R_{k,\text{H}}(\chi))))$. With the average found is then computed standard deviation for both cases, with $\sigma_{S1} = \sqrt{\sum_{k=0}^{At}(\max(R_{k,\text{S}}(\chi)) - \overline{S1})^2/At}$ for the S channel, and for the H channel is calculated using $\sigma_{H1} = \sqrt{\log(1/(\Psi + \text{Y})^2)}$, with $\Psi = (\sum_{k=0}^{At} \sin(\max(R_{k,\text{H}}(\chi))))^2$, $\text{Y} = (\sum_{k=0}^{At} \cos(\max(R_{k,\text{H}}(\chi))))^2$. At this point if $(\sigma_{H1} > 10\%R_{HS} \vee \sigma_{S1} > 10\%R_{HS} \vee (\sigma_{H1} + \sigma_{S1})/2 > 15\%R_{HS})$ then it is concluded that these persons are not the same, otherwise the algorithm continues. Please refer that both H an S values range from 0 to $R_{HS} = 255$. A similar procedure is then performed again, but instead of using all available $At$ segments, are only used the available segments $At$ that also obeys to $\max(R_{h,\{\text{H/S}\}}(\chi)) \in$

$[\overline{\{H1/S1\}} - \sigma_{\{H1/S1\}}, \overline{\{H1/S1\}} + \sigma_{\{H1/S1\}}]$. Using this condition is calculated another average and standard deviation using the same formulas above, obtaining a new $\overline{H2}$, $\overline{S2}$, $\sigma_{H2}$ and $\sigma_{S2}$. Once again, if either $(\sigma_{H2} > (10/2)\%R_{HS} \vee \sigma_{S2} > (10/2)\%R_{HS} \vee (\sigma_{H2} + \sigma_{S2})/2 > (15/2)\%R_{HS})$ the algorithm concludes that the user is not the same, otherwise proceeds to the next step. The last step is to compute an average of all comparisons of all $R_{h,c}$ at the position $\overline{H2}$ for channel H and $\overline{S2}$ for channel S. This average is computed with $Ag_{\mathrm{H}} = 1/At \sum_{k=0}^{At}(R_{k,\mathrm{H}}(\overline{H2}))$ and with $Ag_{\mathrm{S}} = 1/At \sum_{k=0}^{At}(R_{k,\mathrm{S}}(\overline{S2}))$. If $((Ag_{\mathrm{H}} + Ag_{\mathrm{S}})/2 \geq TP \vee Ag_{\mathrm{H}} \geq TH \vee Ag_{\mathrm{S}} \geq TS)$ then these two persons have similar outfit (colour).

- Is performed the exact same method as in Section 5.5.1 - Criterion (b), checking if distances of the skeletons segments are alike.

If these two conditions are true, the user is considered to be the same.

**Global Sound Extraction**

Interaction through sound is still possible even if a user is not detected/presented just in front of the installation. As mentioned in Section 5.5.1, speech recognition can be done by a single Kinect, however in the present 8 Kinect configuration, audio source location was a great part of this range being overlapped, i.e., $8 \times 100^{\circ} = 800^{\circ}$, instead of the $360^{\circ}$.

So the Kinect that is most frontal to the sound source location must be selected. When a user is interacting with the system using their voice or when environmental sound is present, to determine which Kinect will handle speech recognition the follow algorithm was applied:

- Capture the sound beam and confidence levels from all 8 Kinect sensors (individually).

- Sum and store the result of the individual audio beam angles multiplied by the respective confidence levels.

- In order to filter out isolate sound locations, previous step is repeated several times (it was used 10).

- In the end of previous step (all repetitions/sums) is determined which of the 8 Kinects has the greatest sound source locations times confidence level stored.

- In case of a tie or close tie (less than 5% of the highest value), the chosen value will be the one that has the sound beam closest to 0° (corresponding to the most frontal sound in respect to a Kinect position).

- Select Kinect speech recognition from the selected Kinect correspondent to the maximum value calculated.

Fig. 5.9 shows real time sound source beams from all Kinects, in a polar plot. In this case the Kinect selected is the one that is covering the region closest to the 90°. The selected Kinect now works by analysing statically stored keywords that can be detected with a degree of certainty every time a user speaks the keywords, for that the installation in its database has a list of "keyword" and "key-phrases" (linked to a GRXML file) that when identified and confirmed (using the available functions in Kinect SDK) the application triggers a response using a stored answer, using an implemented text-to-speech functionality or optionally a .wav pre-recorder sentence.

The keywords can also trigger actions similar to the gesture interaction. Finally, if there is sound and no word (and no gesture, or user detected) is recognized in the followed 30 s (configurable), then the installation return an audio personalized message to call the attention to itself.

### 5.5.2 Visual Output and Database Module

There are two types of visual outputs visible to the user: interaction through menus in a screen and the hologram exhibition on the top installation (see Figs 5.3 bottom and 5.11 top). Each visual output was built as a single and independent application

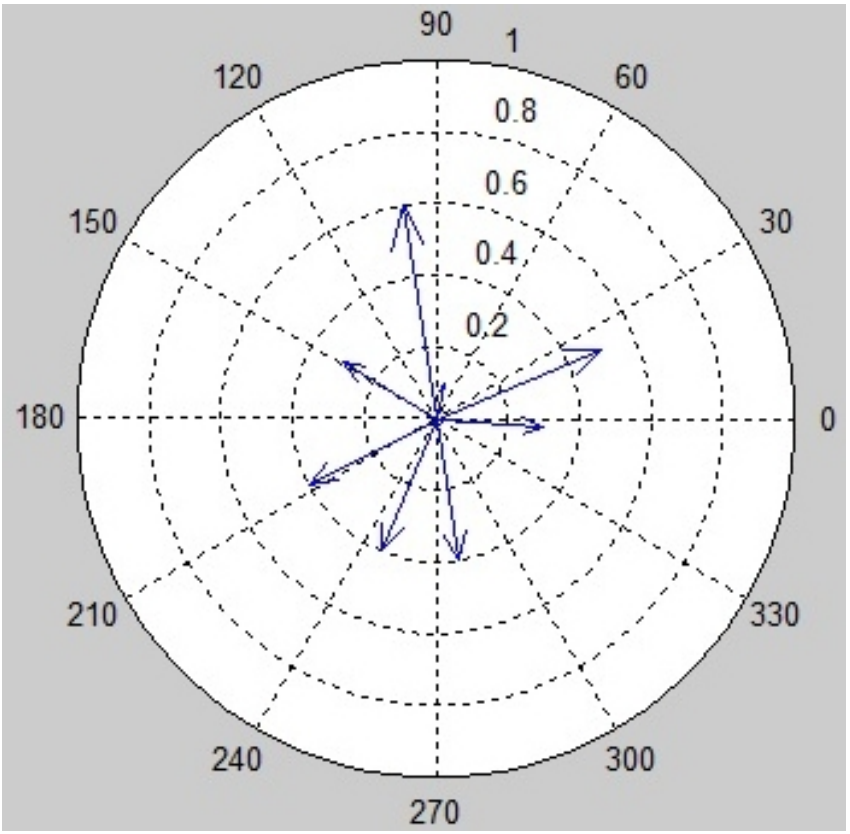


Figure 5.9: Polar plot of sound source beams from all 8 Kinects.

with Unity 3D Engine application (Unity, 2014), with the capability of interacting with each other. Menu interface is presented on a normal screen and the hologram with the projector (see Section 5.4).

The menu application can generate the following layouts: (a) Bridge to other menus containing several titles and options. (b) Similar to a), with the addition of small description for each options if needed. (c) Media content as an image or video with description, useful to show users ideas or video concepts, generating a layout similar to b). The same layout can also be displayed in a diagonal instead of vertical. (d) A menu that only displays images or videos. (e) A menu that enables the user to take his photo and save on the menu.

Both the hologram and the menu applications communicate each other via sockets. When a user appears in front of the installation, the hologram turns to the user and, in case of an avatar, waves (other actions can be triggered by the avatar, the implementa-

tion of that action using Unity is out of focus of this paper).

Database is responsible for storing three types of information: (a) menus configurations, (b) hologram configurations and (c) statistics. Menus configurations are handled by the interface module, generating several types of layouts as soon as the application starts as well as generating responses to the user inputs. Every font type and colour, texts, images and videos of every type of the menus layouts can be customised and saved into the database enabling the administrator of the installation to change all these configurations as he/she intends to. The configurations of the hologram are also saved into the database, allowing the administrator to choose from a variety of different objects or 3D human models, positioning and rotating the hologram around the installation.

All interactions, time spent interacting with menus, as well as requested information of the installation are saved into the database. Although most of these statistics are already available when saved into the database, building heat maps are not immediately saved on it to enable creation of heat maps of groups of users or just a single user. The application saves into the database how much time the interacting user spent seeing a menu, a content and how many times the user have selected them, as well as the user face and full body photo if the user allows to. Being the heat map statistics a special case, is presented in the next session.

**Heat Maps**

With the users global coordinates, is possible to generate heat maps statistics for a specific or a group of users. The generated heat maps are about $11 \times 11$ meters, since Kinects field of view is about 4 m ((Kinect, 2014b)). The combination of all 8 Kinects is a shape similar to a ring (as already mentioned). To generate the heat map a matrix $M$ is created with the size of $m \times n$ px (the default configuration $m = n = 1000$), with each pixel representing 1 cm. Using a group of global positions and starting with matrix $M$ equalling zero, every detection $Pg$ is converted to an auxiliary 2D coordinate $hc$

with $hc(x,y) = (Pg_x \times 100 + m/2, Pg_z \times 100 + n/2)$. Matrix $M$ is now incremented at position $hc$ as well as all neighbourhoods in a radius of 20 cm: $M(x_t + i, y_t + j) = M(x_{t-1} + i, y_{t-1} + j) + 1$ with $\sqrt{(i - hc_x)^2 + (j - hc_y)^2} \leq 20^2$ cm. After this operation, matrix $M$ needs to be normalized to its maximum $h_m$ with $h_m = \max(M)$, resulting a matrix $\dot{M}(x,y) = (M(x,y)/h_m) \times \tau$, with $\tau = 255$. Finally, the map is converted to JET colour space map, Fig. 5.8 right.

## 5.6 Results

The prototype installation has run for several continuous days in the University campus for testing. All algorithms were tested either independently and as a whole application, where no changes on the results were noticed. For each algorithm test, 1 hour of footage was selected for result analyses:

(a) The implemented gestures showed very good results, being very reliable. The gestures detection successful rate was approximately 98%, being too difficult to tell exactly what failed, with this value being calculated using the total succeeded gestures dividing the total attempts (ground truth), both counted in the footage by a human observer. Nevertheless, failures were probably due to users not performing both or either the total distance or speed. Users discovered how to interact with the interface easily, but with the pose gesture being less intuitive than the swipe gesture. Due to this, it was also given the option to use vertical swipe gestures to navigate the menus up and down, instead of using the pose gesture. This change is configurable in the database.

(b) For the user's head direction estimation, results were not as good but reliable enough for this application. The main problem with the algorithm is relying on the eye coordinates provided by Kinect. Although these coordinates are good for the algorithm, they may not be detected as distance to Kinect increases (furthest than 2.5 m these coordinates are hard to be found by Kinect), compromising the algorithm.

However this is true, the results in case a user is close to Kinect were good, detecting successfully the head direction about 87% of the total time counted. It's also important to state that when the user is not close to the Kinect (distance furthest than 2 metres), the successful rate drops significantly to less than 40%. This results are due to the eyes' position not being both detected, occurring when the user is not looking at the centre. If this success rates are not consider enough (in the present case it is more than enough), other algorithms can be considered, e.g. Mora and Odobez (2012), usually they are more CPU demanding.

(c) For the global coordinates conversion, as well as detecting users changing Kinects the results were very good. Kinect only detects users when they are facing the camera, thus only if they are facing the structure. The algorithm worked 100% of the time if the users changed between sensors through overlapping areas as well as making this a gradual change, meaning that the user has to spend about 1 second in the overlapping zone and facing the structure. Due to the overlapping area being small (12°; see Fig. 5.3 and Section 5.4) users can change between Kinect sensors "without" passing in the overlapping area, for instance if they pass in a fast pace. In this last case, the algorithm/procedure could be optimized by increasing the overlapping zones: (i) placing Kinects more closer to each other in direction of the centre of the structure, improving overlapping area although the overlapping angle remaining the same, (ii) use more Kinects to increase the overlapping angle and area. All algorithms stated are prepared to work in these two situations with the only downsides being for (i) the fact that Kinect sensors need to be inside, "near" the centre of the installation obstructing the projector image and (ii) increasing the CPU requirements as well as USB interface ports needed to handle these data.

(d) On the other hand, results on recovering users with obstruction algorithm were satisfactory. The results worked very well recovering 100% of the users if the obstruction doesn't take more than the 5 seconds used and the user does not move too much in that time window. However, some users moved during the test to go to an area

where the obstruction did not occurred, compromising the results. Even in this case, the results are satisfactory, since users can be recovered afterwards with the skeleton recognition algorithm.

(e) The skeleton recognition were tested in several conditions, including in more "extreme" situation, see Fig. 5.10, where persons were recognized entering a bar. The algorithm showed good results for single cameras recognition. In the 1 hour testing, there were about 73 persons passing in front of the installation, with 35 persons being completely analysed. From this group of people, 25 returned to the recognition test site, with all of them being recognized correctly except one of them, see Fig. 5.10 right. As seen on this figure, the person who was incorrectly analysed, had a similar outfit and skeleton with another user. On the other hand, the algorithm for skeleton recognition using multiple Kinects were not as good. Using for it the same site and group of users, except each Kinect now has a different perspective of the site, one focusing an indoor door, the other focusing an outdoor door, from where all persons had to pass. Only 13 people out of 25 were successfully recognized. After this test, and since the results were not as promising as the comparison for the single camera, a repetition occurred in laboratory conditions with lightning condition more closer to the sites where the installation will be used (in indoor events). After this test, it was found out (as expected) that high lightning condition changes between each sensors have a high negative impact on the results, affecting each colour segment differently, thus compromising the algorithm since it analyses a deviation that should be approximately equal to all of these segments. However, the installation is used in indoor events where lightening conditions are approximately constant and lit with electric light sources.

(f) The global speech recognition testing was done in different scenarios. A person located 2 m in front of the installation spoke a series of 100 words in three different noise level backgrounds, from the normal noise level of the room 42.5 dB, to a higher level of noise 57.5 dB and finally 67.5 dB with these two last tests using a pre-recorded

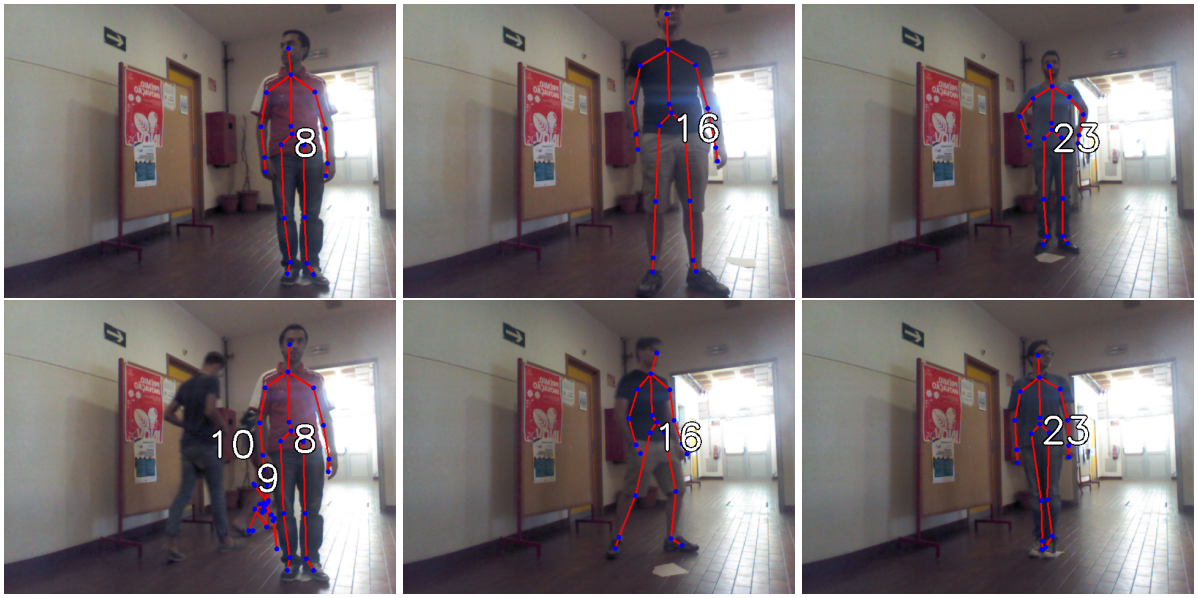


Figure 5.10: Results obtained from recognition using skeleton descriptor.

sound of a crowded bar. For the first test a keyword was successfully detected 94%, while for the second test was 82% and the last test showed a success rate of 53%. The results are expected, since the noise level using the pre-recorded human voice camouflages the words spoken.

(g) Finally, as mentioned in the beginning of this section all the installation were put under stress condition inside the lab, where it stayed connected during several days and where several students and professors were invited to interact with the installation. All systems work well with no problems noticed during those days. Fig. 5.11 shows the prototype installation, in the top row a global view of the installation, where the yellow hologram turn to the person that is interacting with it. Middle row, a detail of the hologram transitions, and in the bottom, different holograms projected. For the best of our knowledge, there's no similar work combining all or most of the features required. For this reason, comparing each of the present algorithms separately with others would not be fair, since all algorithms take advantage of each other by using all previously obtained information, which is used for the statistical purposes."

Figure 5.11: In the top row, a global view of the installation, middle row, a detail of the hologram transitions, and in the bottom, different holograms projected in different views.

## 5.7 Conclusions

It was presented an holographic installation capable of interacting with users in 360° region around the installation. It uses 8 Kinects, enabling a interaction and a holographic host with a upgraded technique of Pepper's Ghost with 8 views. Interaction with the installation is made through gestures and keywords, while creating several statistics about accessed contents and time spent on them, with these contents being highly configurable through a database. The installation is also capable of creating heat maps, estimating the user head position and recognizing users, all available to create statistics on-the-fly for the company to analyse.

Before running, the installation can be configurable to adjust the available content, configuring tests, images and videos and even how the information is displayed. Small details like font type or colour can also be customised. After an exhibition or event, all statistics are displayed on the database for a better comprehension of what products were requested the most, allowing safe changes on the marketing strategies of the company.

Users are tracked and saved into two lists, where a record of all joints are recorded and kept up-to-date. This allows the use of this data for statistics while also allowing analyses of these previous data for inputs. Gestures, for example, are detected analysing this list, allowing (although not done in this case) to read swipes with different amplitudes or velocities, which proved to be very reliable in terms of successful rate. On the other hand, users lost during small fractions of time, due to other users passing in front of Kinect, can be recovered alongside their previous information. Although this algorithm is simple is very useful in this particular application, allowing less number of lost users turning it less redundant. Head estimation is also useful to know if a user was paying attention to the content while interacting, giving a estimation of time a user truly spent looking on.

Skeleton colour-recognition presented to be a very useful and robust algorithm for

the application in hand, enabling recognition of users even if they exit the interaction area and return some time afterwards. These algorithms take advantages of the gathered and processed data from other algorithms where, for example, methods like skeleton descriptors, gestures detection and occlusion detection rely on data gathered for statistical purposes. For this reason and although other algorithms exists on all the subjects above, all algorithms were built to operate in real time with 8 Kinects running at the same time, whereas comparison with said algorithms are out of focus of this Chapter.

With the system very stable, future work will focus on improving users recognition by enhancing the algorithm that recognizes users from different Kinect sensors. Also, descriptors can use other features like symbols or patterns from users clothes to make this improvement. Users Head estimation needs also to be upgraded, despite the fact Kinect resolution being low, and in this matter Kinect 2 should be studied. For the occlusion detection, a early version of the descriptors could also be used to improve the results, using less then the 60 samples used for the normal recognition. Other improvements in consideration is a framework for the creation of menus, allowing more options than the different layouts available at this time. Finally, despite the empirical usability tests done with several users, more experiments should be done in the future to demonstrate the usability of the proposed methods following e.g. the work of (Wei et al., 2015).

# 6

# Augmented reality system to assist inexperienced pool players

## 6.1 Abstract

Pool, pocket billiards or pool billiards, is the family of cue sports and games played on a pool table with six receptacles along the rails called pockets, into which balls are deposited as the main goal of the game. There are hundreds variations of pool games: some of the more popular variations include eight-ball, nine-ball, ten-ball, straight pool, one-pocket and bank pool. PoolLiveAid is an augmented reality tool designed to assist unskilled or amateur pool (snooker, or billiards) players. The system is based

on an HD projector and Kinect 2 sensor placed above the table, acquiring and processing the game on-the-fly, detecting the table's border, balls' position and the pool cue direction in order to compute the predictable trajectories of the balls. The output result is then forwarded to a projector to make it visible onto the snooker playable field.

## 6.2 Introduction

Usually the first contact with a game of pool can be very frustrating for an unskilled player, requiring many hours of practice to understand even the more basic and classical physics involved in this game.

In this Chapter an application is introduced to help and assist mainly amateur players by using the pool table as an interface, showing on-the-fly a prediction of what will happen if the player chooses to hit the white ball as it is, helping the player to make a better move, decreasing the learning curve and preventing him/her from playing countless times before getting it right. Furthermore, a group of menus can be projected and accessed over the table or elsewhere, where for instance a skilled player can save a specific layout of a move (or a group of moves) and load it later, projecting it directly onto the pool table to practice and achieve the best shot possible. Other options are also available, including projection of the game in real-time to a wider screen.

The system was design to work with several varieties of tables and cues, regardless the size, cloth or cue colour and material or even the game type and was based on a Kinect v2 sensor and an HD (or Full HD) projector placed above the table (an extra projector or screen is necessary when the user doesn't want to project the menus and its respective outputs onto the table). To access to the menus a Leap Motion Sensor was used. The Kinect v2 sensor is responsible to capture the game, which is then processed by the standard computer Kinect is connected to, enabling detection of game elements such as table borders, cue direction and balls' position, all used to predict a trajectory. The output result is then forwarded in real time to a projector, showing what might

be the final result of that move onto the pool playable field. With the help of the Leap Motion sensor and very intuitive swipe movements, users can navigate in a set of menus allowing, between other things, access to statistics, replays, real time streaming to a second screen or saving a move layout.

The main contributions of this Chapter is a pool application to help inexperienced players that: (a) works in real club, pub or exhibitions environment, without the need of any changes in terms of table position, lights, etc. (b) Only one support is needed above the table: for the sensor and for the projector. (c) Uses the table as the surface for the projection and interface with the user, (d) has a group of menus projected onto the table (or to a screen, optional) with a set of options that inexperienced or experienced users can select in a way to show or improve their game and moves.

In this section the theme was introduced, in Section 6.3 is presented the state of the art, in Section 6.4 the system specifications and layouts are specified, in Section 6.5 is explained in detail the system implementations, detection of the table borders, balls and cue as well as mapping all outputs to the projector. In Section 6.6 presented and explained the interaction with the menus of the system. Section 6.7 is presented the tests and results of the system done in real conditions (including in exposition environment). In the final Section 6.8 discussion and conclusion are presented as well as future work.

## 6.3 State of the art

There are several examples of tools connected, to some extent, to the game of pool, snooker or billiard. Denman et al. (2003) presented three tools applied to footage from snooker broadcasts. The tools allow parsing a sequence based on geometry, without the need for deriving 3D information, while also allowing events to be detected where an event is characterized by an object leaving the scene at a particular location. This last feature is a mechanism for summarizing motion in a shot for use in a content

based summary. Shen and Wu (2010) also analyses videos, by using an automatic segmentation method of local peak edges to extract the table, and several pre-processing, morphological processing, clustering and HSV colour space the balls are detected to produce a 3D reconstruction of the game. Gabdulkhakova and Kropatsch (2014) also analysis video footage, predicting critical points of the trajectory using structure of a scene and a physical motion model. Also related to video analysis for a 3D representation with different goals work exists Guo and Mac Namee (2007); Höferlin et al. (2010); Legg et al. (2011); Parry et al. (2011); Ling et al. (2012); Jiang et al. (2013).

On a different level, Dussault et al. (2009), Archibald et al. (2010) and Landry et al. (2011) presented a computational system to create a robot capable of selecting and executing shots on a real table. Some of these authors, Leckie and Greenspan (2006) presented a Chapter on the physics of the game of pool (also, about physics of the game of pool see Shih (2014); Shih et al. (2012)). One of these authors has a web page with a tool somewhat similar to ours: ARPool is a projector-camera system that provides real-time feedback to a pool player directly on the surface of the table ARPool (2014). However, to our knowledge, there are no publications on this tool (only the web-page), and the system used a camera as an input device to what is occurring on the table. Also related to "robotic pool", Nierhoff et al. (2011) presented a robot capable of playing on a normal-sized pool table using two arms. The robot can accurately locate the pool table, the balls on the table and the cue, and subsequently plans the next shot. In this case they use a green pool cue and an almost white cloth (they also project trajectories on the table). Both robotic systems were tested under laboratory conditions (specific lightning conditions, etc.).

The present authors presented in Alves et al. (2013) an initial version of the system, very similar to Larsen et al. (2005), using a single Full HD webcam as a sensor to acquire what is occurring on the table. Despite the good results of the system, some limitations occurs, e.g., being very difficult to detect and segregate individually each ball, when a group of two or more balls were connected. Also when using the system

in real pool houses, some limitation were observed on the balls detection due to light limitations in some of those establishments. Shih et al. (2012) presented a system to compute the best sequential shots for a given start cue position. They propose an algorithm to apply maximum tolerance angle search sequentially. The strategy considers combinations among all pockets and target object balls during both the pre and post collision shots selection processes. Later, Shih (2014) presented three novel gaming strategies to investigate the effect of cue shots planning on gaming performance. The simulations were conducted based on a collision model considering the restitution effects. An augmented reality training facility was devised to guide users in both aiming and cue repositioning control in a real-world billiard game.

## 6.4 System specification and layouts

As already mentioned in the Introduction, Section 6.2, the system has five major components: a pool table, any dimension, with the usual balls and cue, a Kinect 2 from Microsoft (Kinect, 2014a), any ordinary laptop or desktop computer capable of analysing inputs from Kinect, a projector to project the computed trajectories and balls' locations and a Leap Motion sensor to interact with the menus projected onto the pool table or other screen. Is important to refer that when written "any dimension", a single Kinect 2 can cope with tables up to $2.50 \times 1.4$ m (meters), requiring two or more Kinect 2 sensors to acquire the entire game field for tables with higher dimensions, implying an additional algorithm to merge all acquired images from all used Kinects.

Fig. 6.1 left, shows the system layout, with the position of Kinect (marked with number 1) being more or less the centre of the table. Other positions could be used, nevertheless, due to the distance limitations of Kinect, better precision is achieved when the sensor is placed at the centre of the table. The distance from the sensor to the table should be the necessary for the sensor to capture information of exactly the whole table. Fig. 6.1 right shows the Kinect 2 sensor (black) on a white support

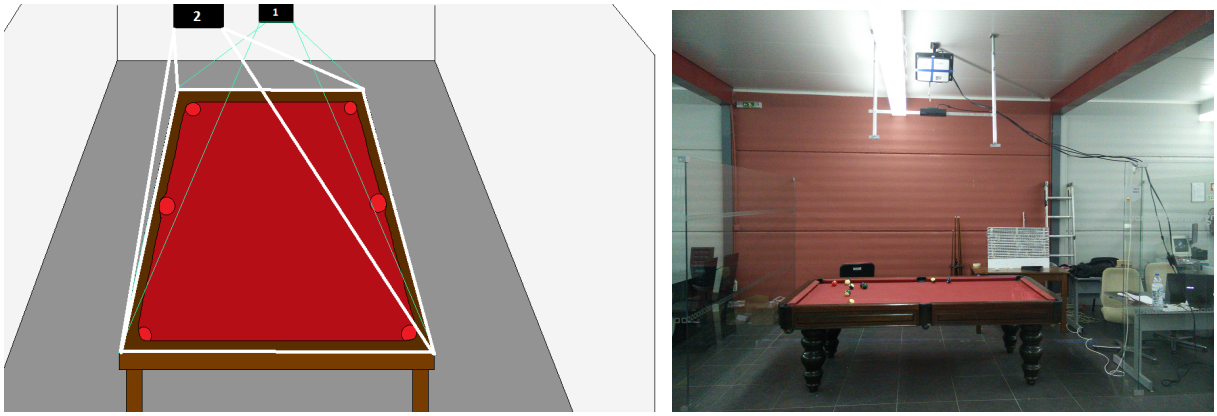


Figure 6.1: On the left, the system layout with the Kinect and projector position relative to the pool table. On the right, the prototype.

with a distance to the table of 1.6 m. Please note that instead of using the Kinect 2, other sensor(s) that returns a depth map could be used as long as the depth resolution has enough precision to differentiate the distance between the balls and the table (less than 2 cm), with for instance on the ends of the support can be seen a previously tested position of two Kinect 1. All the algorithms presented in this Chapter can also be used with those devices being necessary one additional concatenation algorithm of the two depth frames into a single one. For the projector any specifications can be used, as usual, taking only into account the lightning conditions of the specific place of the system, with more ANSI Lumens needed if the projection occurs in an exposition with less being necessary in a pub and with a better resolution meaning a better visualization of the results. The projector can be placed in a side hall or over the table, as long as it can project onto the whole table, Fig. 6.1 left (mark with the number 2), or in the right, fixed in the celling.

## 6.5 System implementation

Although the computer application must obey the game rules, it can also take advantage of them in order to improve the reliability of it. For a pool/snooker/billiard game, among other rules, some are coincident and very important: (a) Every action

takes place on the table; (b) Players can only make a move when all balls are stopped and in their final position; (c) Players can only play the white ball and cannot interfere with other balls directly; (d) Only one player can make his move at a time, meaning, only one cue can be in the playable area of the snooker table.

In the following section the framework will be detailed.

### 6.5.1 The framework

The Kinect sensor provides, among other things, an RGB frame denoted $I(x,y)$ and a depth frame $D(x,y)$, where each pixel $(x,y)$ represents a quantified distance of each point of an object or surface to the camera. A blacker pixel means the point is closer to the camera, while a whiter pixel means the point is further from it. Since Kinect depth frame has a 16 bit resolution, a pixel value can range from 0 to 65535 (the last value is represented by $N_D$). All the 16 resolution bits of the depth frame were used.

Fig. 6.2 shows the global flowchart of the implemented software, following the pool rules (see above), in order to detect all elements involved in a game. A new depth frame $D$ is obtained from Kinect each $\Delta t = 1/30$ s (seconds). In each interval the algorithm analyse the depth frame $D$ to know in what stage the game currently is: (a) If the motion/game play has stopped, meaning that "no motion" was detected in the current instance but "motion" was detected in the previous instance, then it must *detect and classify the balls* but, (b) if "motion" was detected on the current instance and the previous instance, then two situations can occur: (b.1) the *balls are in movement* on the table, or (b.2) a sudden movement was detected, assuming that a player is making his move with, in this case, the balls already detected in a previous run, and a *cue is in the table*. Again two situations can occur: (b.2.1) *the cue detection* should be computed as well as physics, or (b.2.2) *there is a strike* meaning balls are still moving and need to be detected once they stop.

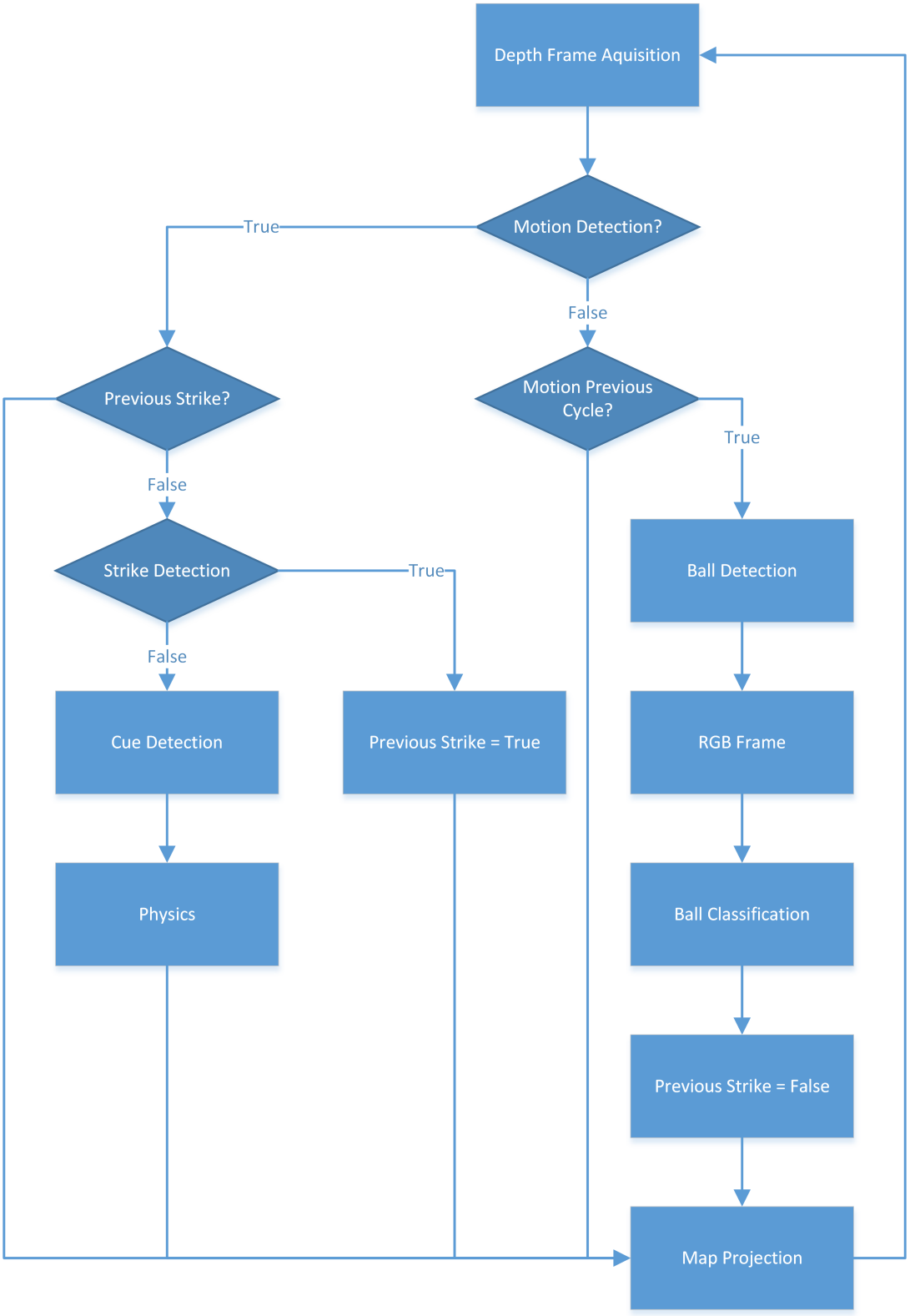


Figure 6.2: Flowchart explaining the procedure of the algorithm implemented.

### 6.5.2 Auxiliary processing steps

Before starting detailing the system, three important auxiliary processing steps, used in almost all algorithms, must be introduced: (i) computing the perspective transform of frames acquired by the sensor, (ii) computing the perspective transformation of images that will be displayed by the projector, and (iii) the reduction of inconsistencies on each depth frame. Most of the system steps enunciated in Section 6.5.1 and detailed below only compute information over the playable area of the game, both the Kinect sensor and the projector capture and project, respectively, more than that. Also, as mentioned in the system implementation, neither the Kinect or the projector need to be placed in the centre of the table, but to achieve better results is recommended for both to be placed as shown on Fig. 6.1.

One of the auxiliary processing steps needed was the (i) perspective transform on the sensor. By using the RGB image $I$ from the Kinect, a Canny edge detector (Russ, 2010) was applied, and after an Hough transform (Russ, 2010). Selecting only the (almost) horizontal and vertical lines, the interception points between them were computed and presented to the user (see also Alves et al. (2013)). The user is asked to validate those corners, adjusting if necessary due to small errors implying larger trajectory imperfections, with $Ct_{\{1,\ldots,4\}}(x,y)$ the positions of the corners on the playable field from top-left in clockwise order. Fig. 6.3, first row left, shows one example of the above configurations.

Using the four points computed above and the 4 point where the mapping needs to be translated to, i.e., the four transformed coordinates, respectively $(0,0)$, $(M,0)$, $(M,N)$ and $(0,N)$, with $M = 2 \times N$ and $N = (d(Ct_2,Ct_3) + d(Ct_1,Ct_4) + (d(Ct_1,Ct_2) + d(Ct_4,Ct_3))/2)/4$, with d the Euclidean distance, then a transformation matrix $M^{PtK}$ can be computed. Using this transformation, the initial depth frame $D$ can now be transformed to a depth frame containing only the playable field, $D' = M^{PtK}D$. The reason for $M = 2 \times N$, is that a professional pool table, have a width 2 times larger than its height (if chosen a table that doesn't obey this criterion, then the respective

proportion must be considered; this can be adjusted if necessary in the initial configuration menu, see also Section 6.6).

Some projectors may have a built in function that lets the user choose the four corners of the projection, enabling the projector to compensate and distort the output image in order for that image to fit those corners. Since not all projectors have this built in function (ii) a similar process to (i) is necessary. A white polygon on a black background (Fig. 6.3, first row right) is projected onto the table (Fig. 6.3, second row left). The user then selects and adjusts the four corners until the polygon cover the whole playable area. Those correspondent points, positions of the corners of the projector field from top-left in clockwise order were $Cp_{\{1,\ldots,4\}}(x,y)$. A perspective transformation for the projector, $M_{PtP}$, must translate exactly the coordinates of game elements found, on a transformed depth frame, to the projector field. Now the four point of the table $(0,0)$, $(M,0)$, $(M,N)$ and $(0,N)$, where mapped to respectively $Cp_{\{1,\ldots,4\}}(x,y)$. The computed frame compensating projector distortion and containing the elements (e.g., trajectories , menus) $P$, can now be transformed $P'' = M^{PtP}P$.

The computation of the above transformations matrix $M^{PtK}$ and $M^{PtP}$ is done only once in the initial configuration of the system, where the matrix coefficients are saved into a file, which is loaded every time the system initializes.

(iii) Any depth frame acquired by the Kinect presents small inconsistencies and noise, between $\Delta t$ frames, that need to be removed in order to improve detections and reliability. Two different algorithms, depending on which phase the game is, were used (see Fig. 6.2): (iii.a) No motion occurs (used e.g. before balls detection), in this situation a small delay (less than $1/2$ s) on a detection is not important, as this delay is not a problem for the player, since balls just stopped moving with the player normally thinking his/her next move. In this case it was used an average of the most recent frames (of course, this can only be used if no motion was detected in all frames). The average filter in instance $t$ is simply the sum of previous $N_P$ depth frames acquired in that time interval divided by the number of all frames summed,

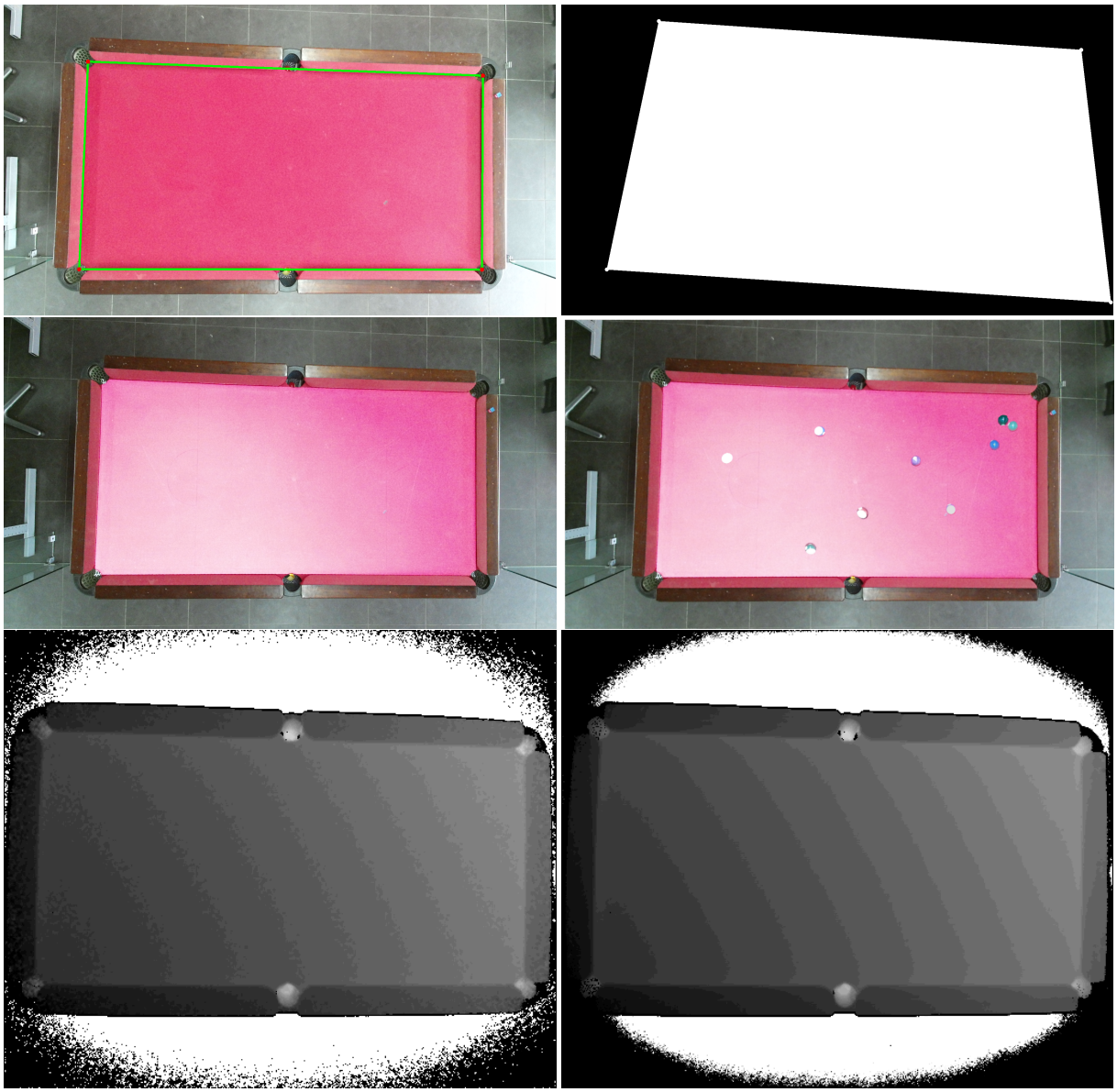

Figure 6.3: In the first row left a colour frame $I$ retrieved with Kinect, with the 4 corners (and lines) detected, the white polygon (right) and the same polygon projected onto the table (second row left). On the second row right the table with balls. Third row, a frame $D$ (left) before and after the application noise removal $Da$ (right).

$Da_t(x,y) = 1/N_P \sum_{k=t-N_p}^{t} D_k(x,y)$, with $Da_t$ being the average frame in instance $t$. The result is represented in Fig. 6.3, third row right, (with $N_P = 10$) for the original image on the left. Choosing a higher $N_P$ improves the reliability but increases the delay, thus the chosen value allows to remove most inconsistencies.

(iii.b) Motion occurs (e.g. before the cue detection), the noise removal has to be done in real time (between frames, less than $\delta t$ seconds). In this case, a Gaussian filter (**G**) (Russ, 2010) was used with $\sigma = 2$, $Dg_t(x,y) = \mathbf{G}(D_t(x,y))$.

One important note, for better comprehension of the remaining text, any frame with the notations $X'$ or $X''$, represents frames were the $M^{PtK}$ or the $M^{PtP}$ transformations was applied respectively, and for better visualization purposes, all pixels from the figures representing the depth $D$ frame retrieved by Kinect were divided by 16, cramped any values higher than 255, then brightened up by 10% and the contrast risen by 90%.

### 6.5.3 Motion detection

Motion, and the type of motion determines in what phase the game is and what step of the algorithm should be done next (see Fig. 6.2). Motion detection can be essentially implemented using the difference between two depth frames ($|Dg'_t - Dg'_{t-1}|$). The problem with this method is that small differences between two consecutive images are almost undetectable (zero) and can be confused with noise.

To avoid this problem and instead of comparing the two most recent frames, a comparison between multiple frames with the current frame was implemented, $M_t(x,y) = \sum_{j=0}^{N_p} |Dg'_t(x,y) - Dg'_{t-j}(x,y)|$, with $M_t$ the motion detection frame in instance $t$, between the current frame $Dg'_t$ and the previous $N_p$ frames, with $N_p = 40$, $Dg'_{t-j}$ (corresponding to 40 frames, around 1.5 s). Nevertheless, as some of those values will still reflect noise, a threshold was applied and a binary image where motion exists was computed, $Mb_t(x,y) = 1$, if $M_t(x,y) \geq T_m$, otherwise $Mb_t(x,y) = 0$, with $T_m = 0.05\% N_D$ (for the $N_D$ value see Section 6.5.1; with the value 0.05% being empirically deter-

mined). Fig. 6.4 first row left, shows the example of a $Mb_t$ frame (with a cue and a ball strike).

To categorize the motion events, the system has a binary variable $\omega$ that states the stage of the system, "motion"(1) or "no motion"(0). The systems starts by default with $\omega = 1$. The amount of white pixels detected give us the information if any motion is occurring on the current frame (as well as some noise), $Cm_t = \sum_{x=0}^{M} \sum_{y=0}^{N} Mb_t(x,y)$. A counter, $K_t$, was implemented to manage false movement detected by the procedure above.

If the current value of the system state is no motion ($\omega = 0$), then is necessary to detect when motion starts to occur, and for this the counter $K_t$ is incremented if $Cm_t$ is higher than $K_1 = 0.05\%M$, otherwise it is decremented (values lower than 0 are cramped to value 0). If the counter $K_t$ reaches $K_2 = 25$ (corresponding to 25 frames, around 1 second), then is considered that motion has started, changing the state to "motion" ($\omega = 1$; $K_t = 0$).

In the other hand, if there system state is motion ($\omega = 1$) then the counter is incremented if $Cm_t$ is lower than $K_1 = 0.05\%M$, otherwise it is decremented (again, values lower than 0 are cramped to 0). If the counter $K_t$ reaches $K_2 = 25$, then is considered that motion has stopped, changing the state to "no motion" ($\omega = 0$; $K_t = 0$). Now combining with $K_t$ the motion can be characterized as follows:

(a) *Stopped*: If motion was not detected on current instance, but was detected on the previous instance, $\omega_{t-1} = 1 \wedge K_t = K_2$, then $\omega_t = 0 \wedge K_t = 0$.

(b) *Started*: If motion was detected on current instance, but was not on the previous instance, $\omega_{t-1} = 0 \wedge K_t = K_2$, then $\omega_t = 1 \wedge K_t = 0$.

(c) *Non-existent* (no motion): If motion was neither detected on the current and previous instance, $\omega_{t-1} = 0 \wedge K_t < K_2$, then $\omega_t = 0$.

(d) *Motion* (in motion): If motion was either detected on the current and previous instance, $\omega_{t-1} = 1 \wedge K_t < K_2$, then $\omega_t = 1$.

Having the motion characterized, it is now possible to detect the balls positions

and the position of the white ball.

### 6.5.4 Ball detection

Ball detection is triggered after motion has *stopped*, and uses a reference depth average frame $Ra'(x,y)$ (for average frame procedure, see Section 6.5.2) taken in the initialization step. That reference frame is nothing but a frame of the empty snooker table (Fig. 6.4, second row left).

This reference frame is used in this step to detect the balls' positions on the table, being $B_t(x,y) = Ra'(x,y) - Da'_t(x,y)$, with $B_t$ the frame containing the blobs where balls might be, Fig. 6.4 third row left, the reference frame $Ra'$ (left) and a current frame $Da'_t$ (right) in the second row right (the original correspondent RGB image can be seen in Fig. 6.3, second row right). Note that an absolute subtraction is not need since the depth frame, returned by Kinect, contains lower values for pixels closer to the camera, meaning that the current frame $Da'_t$ will contain the same values as $Ra'$ except where balls are or some noise occurs. A binary threshold is then applied to obtain a binary frame containing the ball blobs, $Bb_t(x,y) = 1$, if $B_t(x,y) > T_b$, 0 otherwise, with $T_b = 0.05\%N_D$ (Fig. 6.4, third row right). This threshold can be adjusted to detect a higher area of the ball, Fig. 6.4 top row right. If $T_b = T2$ then balls will be detected with a higher area, but on the other hand if balls are closer to each other then they will be forming one blob, thus being only recognised as one ball. For this $T_b$ needs to have a higher value, shown as $T1$ in Fig. 6.4 top right, in order for the balls to be recognised separately even if they are in contact to each other.

The next step consists in applying the morphological operator, erosion (**E**), which has the goal of removing any remaining noise that still persisted (small blobs if still exists) $Be_t = \mathbf{E}(Bb_t)$ (Fig. 6.4 bottom row, left).

Any spherical ball, has a peak in the depth frame in its centre (Fig. 6.4 top row, right) those peaks present the exact position $(x,y)$ of the ball on the table. To get those peaks, a contour finder (**C**) (Suzuki et al., 1985) was applied, $Bc_t = \mathbf{C}(Be_t)$, after which,

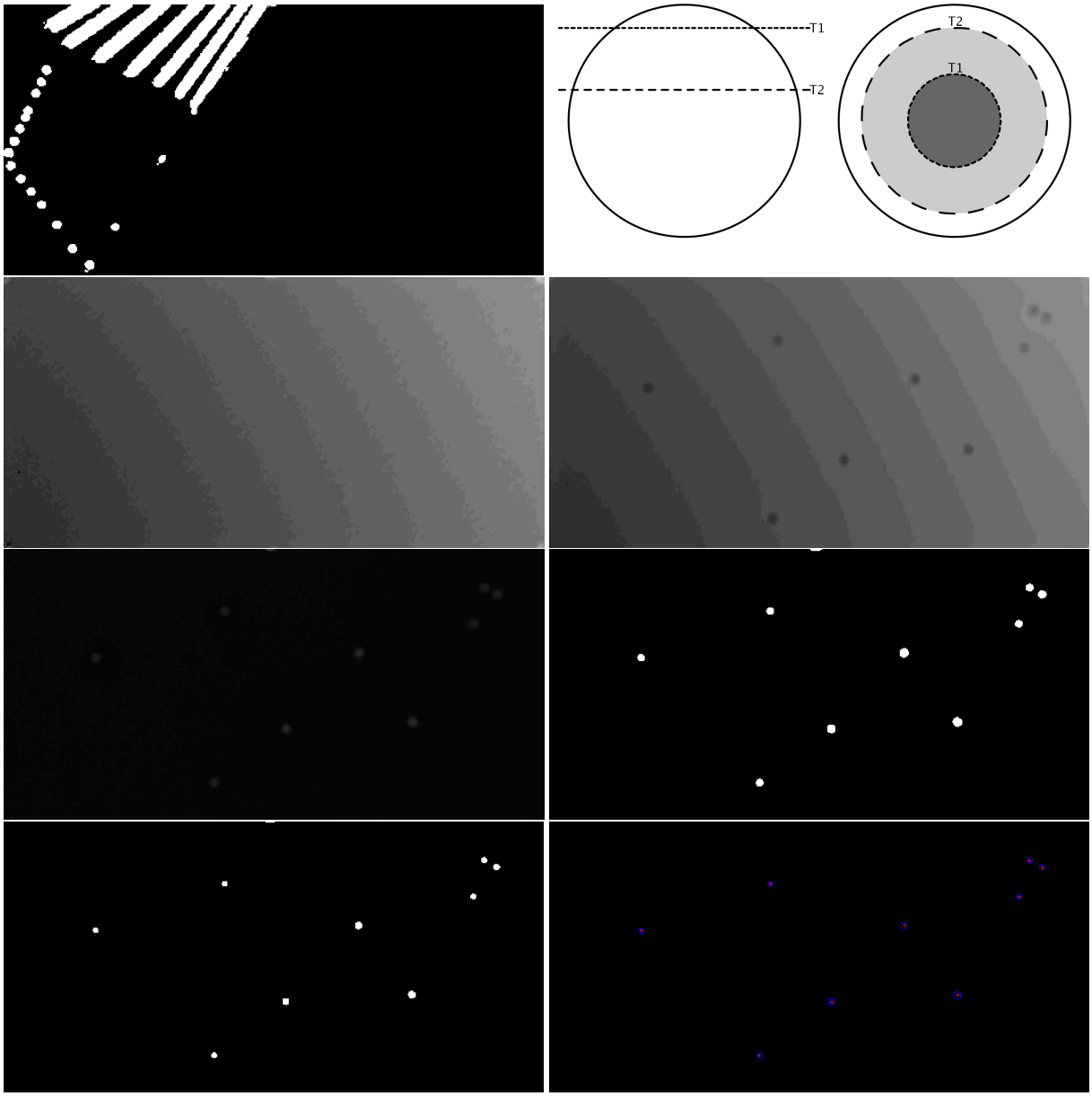


Figure 6.4: From left to right, first row, a $Mb$ frame example, side and top view of a ball, with T1 and T2 lines, representing the area of highest peak of the ball and the relative position of the ball detection contour (see text). Second row, on the left, the reference frame $Ra'(x,y)$ with the table empty (see original correspondent RGB in Fig. 6.3, first row left), on the right an example of a frame containing the balls $Da'_t$ (see original correspondent RGB in Fig. 6.3, second row right). Third row, on the left shows $B_t$ containing the blobs where possible balls might be, on the right, $Bb_t$ binary image with the balls positions. On the fourth row, left the $Be_t$, image after been applied the erosion, and on the right the $Bc_t$ image which has contours marked in blue and the balls' centre in red.

and for each contour a local maximum was computed in $B_t$, corresponding to the ball centre coordinates (this last process has better results than finding directly the centre of the contour). Fig. 6.4 bottom row left, shows in blue the contour of each ball and in red its peak.

#### White ball classification

After each ball is detected, the white ball has to be classified since it is the only ball that the cue may hit. With this purpose, the $I'$ colour frame acquired from the Kinect, was converted to HSV colour space, $I'_{HSV}$, and the pixels from the component value ($I'_V$) inside each $\xi_i$ contour existing in $Bc_t$ were summed. The contour/blob with the biggest sum was returned as the white ball position $(x,y)$ and the respective contour index $i$, $W(x,y,i) = \max_{\{i=0,\ldots,N_b\}}\{\sum_{\xi_i} I'_V(x,y)\}$, with $N_b$ the number of ball contours in $Bc_t$.

A similar process could be used to classify other balls by colour (red, black, blue, etc.), turning possible automatic score points of a game for each player (not the goal of this application).

### 6.5.5 Playing a shot

With all balls detected and the white ball found, cue detection is the next step, as well as strike detection, turning possible the prediction of a strike.

#### Cue detection

When motion stops and triggers ball detection, a reference frame $Qa'$ (after been applied average smoothing; see Section 6.5.2) is taken, see Fig. 6.5 first row left. Since the reference frame contains the balls at the instance that motion stops, the difference between it and any current frame (Fig. 6.5 first row right), can only be a player, a cue or both. Following this, cue detection can be done by $C_t(x,y) = Qa'(x,y) - Da'_t(x,y)$

(Fig. 6.5 second row left). As in the case of ball detection, a threshold is then applied, and a binary frame is created, removing small inconsistencies due to noise, $Cb_t(x,y) = 1$ ,if $C_t(x,y) \geq T_c$; 0, otherwise, where $T_c = 0.05\%N_D$ (see Fig. 6.5 second row right).

Using again Suzuki contours finder (Suzuki et al., 1985) on $Cb_t$, all blobs contours are found ($\gamma_i$), one being the cue (usually with the hand/arm attach) and all the others being noise. If the cue exists on $Cb_t$, then it has a larger area than the rest of the blobs found, Fig. 6.5 third row left.

First is found the blob with the largest area, then computed the average area of the remaining blobs: $A_l = \max(A_i)$, with $A_l$ being the area of the largest blob, $A_i$ the area of each blob $\gamma_i$ in the frame $Cb_t$, and $N_c$ being the total number of blobs, then the average area of the remaining blobs $\overline{A}$ is found, $\overline{A} = (\sum_{i=0}^{N_c}(A_i - A_l)/(N_c - 1)$.

The next step consists in removing from $Cb_t$ all blobs $A_i$ which area is less than $100 \times \overline{A}$, returning an image with only what might probably be the cue $Cc_t(x,y)$. At this point if exists a blob at frame $Cc_t(x,y)$ it is possible to be a hand, a cue or more probably a cue with an hand, Fig. 6.5 third row right.

To detect the cue and later it's direction, the Hough line transform (Russ, 2010) is used on $Cc_t(x,y)$ to find the lines that differentiate the cue from a possible hand. All lines that both starting and ending point are furthest than 5 times the ball diameter ($2 \times r$, with $r$ the ball radius) are discarded, removing this way possible lines detected on the user's hand, with the lines visible in Fig. 6.4, fourth row left. All lines with the same angle ($\pm 5^o$) are then summed to find the average line (Fig. 6.5, fourth row right).

**Shot prediction**

With the white ball located and the cue detected, predicting a trajectory of the next strike is possible. Since the main goal of this application is to help inexperienced players, calculations made of what might be the next trajectory take into account that the player will hit the centre of the white ball. Nevertheless, work that studies the

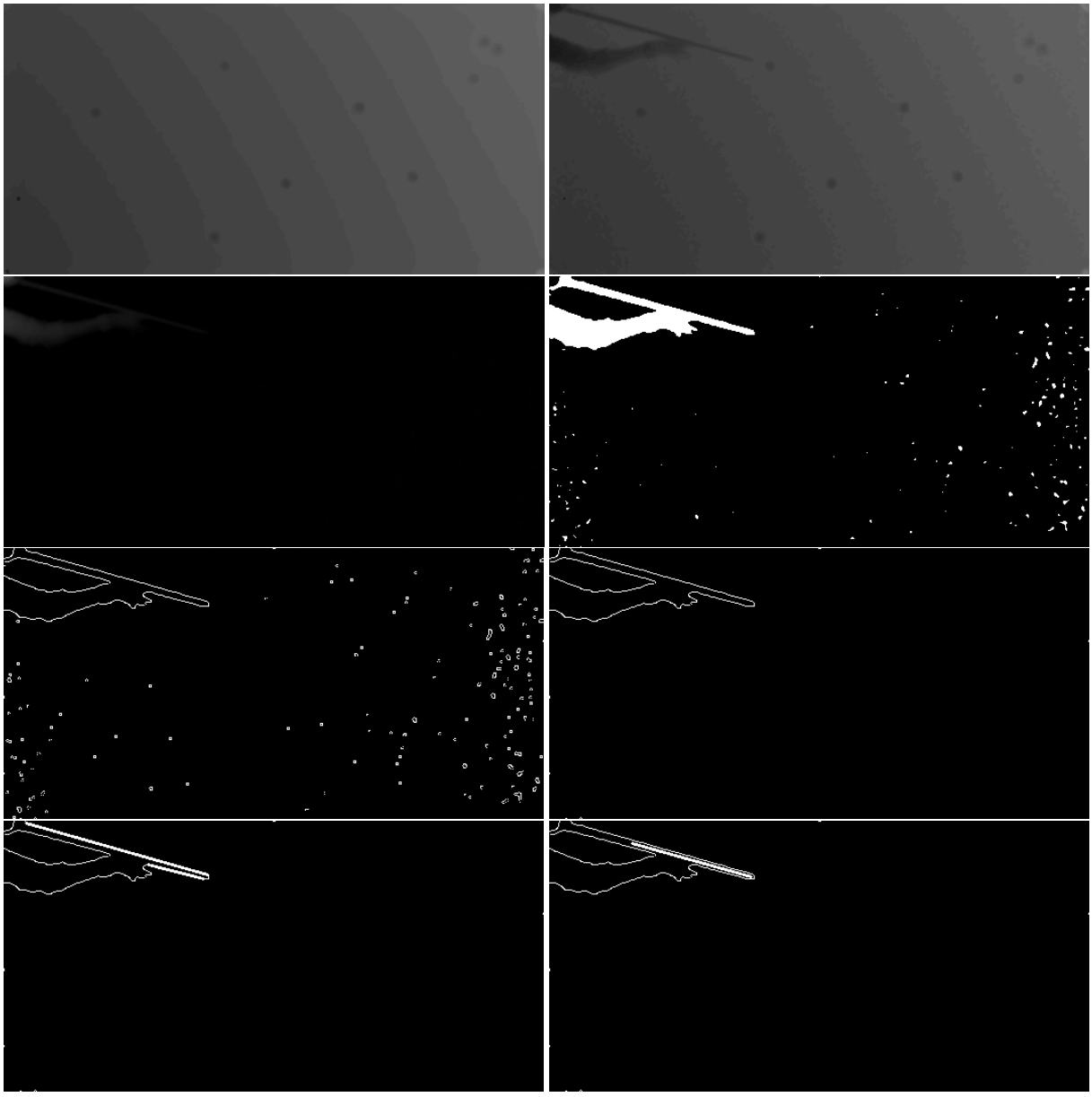

Figure 6.5: From left to right, first row, the reference frame for cue detection $Qa'$, an example of a current frame containing the player hand and the cue $Da'_t$. Second row, $C_t$ and $Cb_t$. third row $Ce_t(x,y)$ and on the right frame with blobs due to noise removed. Fourth row, result on multiple lines found on the cue and on the right single one resulting from the cue detection (for details see the text).

physics when the strike in done in other locations on the ball exists, see e.g. Leckie and Greenspan (2006); Shih et al. (2012); Shih (2014). Before any prediction is made, is important to check if the player is aiming at the white ball and only if this is true a prediction can be made.

To check if the cue is being aimed at the ball (see Fig. 6.6) is computed: (a) the equation of the cue line, $\overrightarrow{v}(x,y) = (x_c,y_c) + k_c(c_x,c_y)$, where $(x_c,y_c)$ is a point on the line, and $(c_x,c_y)$ is a direction vector of the line, computed from the two points found on the cue detection algorithm, Section 6.5.5. (b) The ball is represented by a circle $r^2 = (x - x_{bc})^2 + (y - y_{bc})^2$, with the $(x_{bc},y_{bc})$ the centre of the white ball, and $r$ the ball radius. To check if they intersect, a value of $k_c$ must be a real number solving the equation $(x_c - k \times c_x - x_{bc})^2 + (y_c + k \times c_y - y_{bc})^2 = r^2$.

If $k_c$ is a real number, then the cue is aiming at the white ball and physics can now be calculated. Knowing the vector representing the direction of the cue ($\overrightarrow{v}$), is necessary to compute the vectors of each table boundary in order to calculate the respective reflection trajectory (see Fig. 6.6). Taking in consideration (obviously) that the centre of the ball, due to it's radius ($r$) never touches the table boundaries, the boundaries vectors are computed as follows, clockwise around the table, $\overrightarrow{b^i}(x,y) = (x_b^i,y_b^i) + k_b^i((y_b^i - y_c),b_y^i)$, with $i = \{1,...,4\}$ and $(x_b^1,y_b^1) = (r,r)$, $(x_b^2,y_b^2) = (M - r,r)$, $(x_b^3,y_b^3) = (M - r,N - r)$, and $(x_b^4,y_b^4) = (r,N - r)$. The directions are respectively, $(b_x^{\{1,3\}},b_y^{\{1,3\}}) = (M,0)$ and $(b_x^{\{2,4\}},b_y^{\{2,4\}}) = (0,N)$.

Finding if an intersection exists, and getting the contact point, for every boundary $i$, is then calculated using $(x_c,y_c) + k_c^i(c_x,c_y) = (x_b^i,y_b^i) + k_b^i(b_x^i,b_y^i)$, i.e., $k_b^i = (c_x(y_b^i - y_c) - c_y((x_b^i - x_c))/(b_x^i c_x - b_y^i c_y)$. An intersection between the boundary and the cue trajectory only occurs when $(b_x^i c_x - b_y^i c_y) \neq 0$, i.e., $(c_y/c_x) \neq (b_y^i,b_x^i)$.

If the intersection occurs, the possible contact point $p^1$ to $p^4$ (with $(p_x^i,p_y^i)$ the co-ordinates of contact with the table boundary) in the table boundary are calculated by $(p_x^i,p_y^i) = (x_b^i,y_b^i) + k_b^i(y_b^i - y_c)$ (see Fig. 6.6).

Since the line of the cue can intersect more than one boundary, the true physical

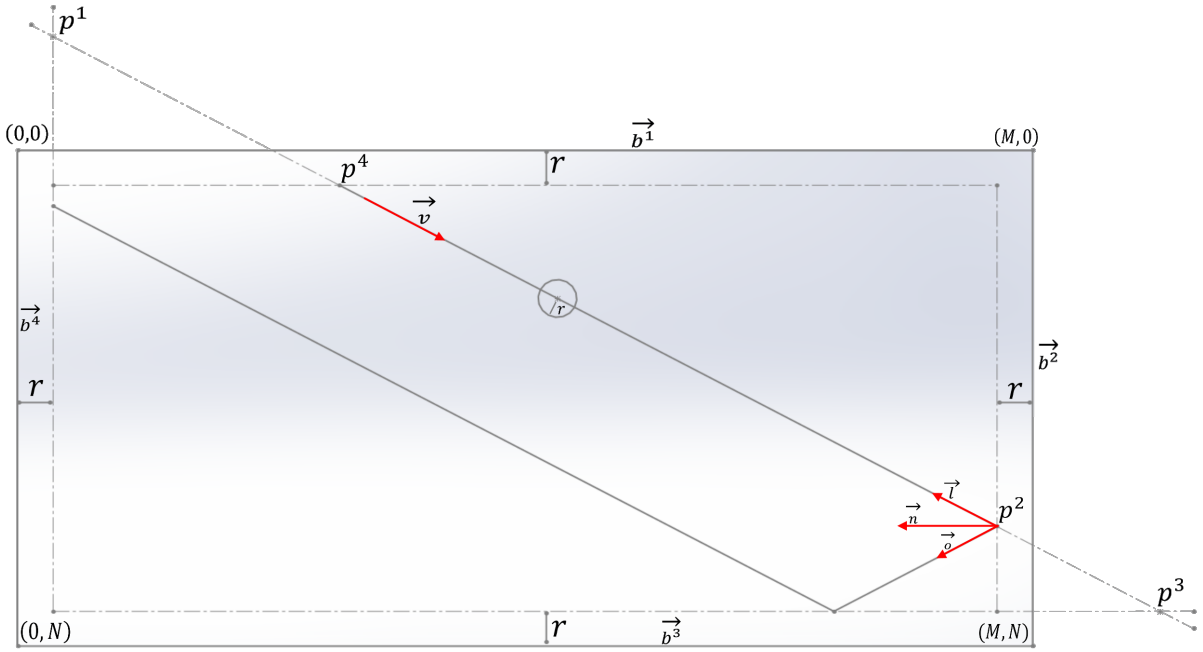


Figure 6.6: Shot prevision, trajectory computation.

intersection needs to be found. Between all boundary's intersection point, the nearest point of contact to the ball that also has the same direction of the vector $(c_x, c_y)$ is found. The point where the contact will occur $(x_f, y_f)$, is the one that obeys $(x_f, y_f) = \min \mid (p^i_x - x_{bc}, p^i_y - y_{bc}) \mid \wedge((c_x \times b^i_x) > 0) \wedge ((c_y \times b^i_y) > 0)$.

The reflection between the boundary and the current direction also needs to be calculated, in case a prediction of a following trajectory needs to be computed. Using the selected boundary $f$ by the process above, a normal vector to that boundary is calculated (see Fig. 6.6), $\overrightarrow{n}(x,y) = \overrightarrow{v}(-b^f_y, b^f_x)$ as well as the vector with the opposite direction to the current trajectory ($\overrightarrow{v}$), $\overrightarrow{l} = (x_c - x_{bc}, y_c - y_{bc})$. The reflection trajectory is then calculated by $\overrightarrow{o} = 2 \times \overrightarrow{n}(\overrightarrow{l} \cdot \overrightarrow{n}) - \overrightarrow{l}$.

This process can be repeated any number of times to obtain more reflection predictions. In the images presented (see Fig.6.7), the process was performed three times in the first row and four times in third row left.

### Strike detection

With a predicted trajectory computed, strike detection is necessary to detect when the white ball starts moving in order to stop calculating trajectories and wait until the balls are stopped, triggering their detection. The detection of movement by the white ball is made again using the reference image $Qa'$ (extracted from $Da'$, exactly when the movement *stopped*), extracting a frame $\tau$ consisting in $Qa'$ with a region of interest (RoI), centred in the current position of the white ball ($W$) and with the size $d \times d$ (being $d = 2 \times r$), the white ball diameter.

For every new depth frame $Da'_t$, a RoI centred in the coordinates of the white ball and with $\tau$ size is applied obtaining $v$. A subtraction is then processed, $S(x,y) = \tau(x,y) - v(x,y)$, and small threshold is applied to remove any noise, $T_s = 0.01\%N_D$, returning $Sb(x,y) = 1$, if $S(x,y) \geq T_s$; 0, otherwise.

Summing all pixels in the image $Sb$ determines if the ball has left its place, $C_S = \sum_{x=0}^{d} \sum_{y=0}^{d} Sb(x,y)$, with $C_S$ being the total number of pixels and representing how much the white ball has left its position. If $C_S > 1/3 \times \pi \times r^2$, then is considered that the white ball has been shot.

### Mapping projection

With all game elements found, is now possible to project computed information onto the table. For that, an image $P$ is used and created with several options (Fig. 6.7): (a) white circles centred in each ball detected, with radius $r + 5$ px (pixels), not filled and line width of 3 px (Fig. 6.7, second row right); (b) a white filled circle centred in the white ball (Fig. 6.7, third row left) or (c) a circle centred in the white ball with line width of 10 px (Fig. 6.7, top row); (d) all the white lines 3 px width, computed from the predicted trajectories $\overrightarrow{o_j}$, with $j = \{1,...3\}$ (all colours used for the projection, as well as dimensions of the circles, lines and other effects, can be adapted customised according to the user preferences). The projector transformation is applied to $P$, see Section 6.5.2, returning the projected image $P''$.


Figure 6.7: Examples of the application working in different expositions and sites.Bottom, right menu example.

## 6.6 Interfaces

In the previous version of the system (Alves et al., 2013), menus were always projected onto the table with the user being capable of interacting with it by analysing RGB frames acquired from the Full HD webcam, detecting simple hand movements to navigate them. Despite many users finding this appealing, others showed preference that the interface and projection could be done elsewhere, for instance navigating through the menu while the game is undergoing or showing a top view of the game in real time into another screen or even showing a replay of a move (this were features that interested more the experience players.)

To include the possibility of interacting with the menus anywhere, two solutions where considered: (a) using the Kinect to navigate the menus when they are being projected onto the table or using a keyboard, a mouse or a touch screen when the projection occurs elsewhere. (b) Using a single sensor that can be used in any situation and doesn't significantly increase the cost of the application.

The second case (b) was chosen, with Leap Motion (Motion, 2014) sensor being the one selected for the task, having an API capable of detecting multiple hand gestures. All menus were developed based on swipe gestures, being more intuitive for most users, reacting to two pairs of opposite types of swipe gestures: (i) Up and Down (select/deselect): is performed by a top to bottom swipe or a bottom to top swipe respectively. (ii) Front and Back (change menu options): is performed by a front to back or a back to front swipe, to select "lower" and "upper" menu options respectively. The implementation details is out of the focus of the Chapter, for those see Sousa et al. (2014); Cardoso et al. (2015). Examples of options available are: (a) play with help, (b) a reload of a previous play, (c) save a game/play, (d) show the game in real time and (e) statistics. Fig. 6.7 bottom right shows a menu projected onto the table. In this image the Leap Motion sensor is over the table for illustration proposes, with this being not recommended since the IR of the Leap Motion sensor causing interference with Kinect

IR increasing noise. If the goal is to have menu interaction only in the table region, it is advised to use the Kinect sensor, the implementation of swipe gestures can be seen in Alves et al. (2014, 2015a).

## 6.7 Tests and results

Fig. 6.7 shows some examples of the system working in real crowded environment on three different expositions, with a duration of 1 week each, where any player could try it. The only limitation imposed for testing to users was basics rules (see Section 6.5). All the tests and statistics were done during this weeks with real users (first time application users and all users were unknown to the development team). Before playing, users were asked if they were beginners or professional and, in case they were professional, statistics would not count although their opinion noted in order to improve the application for professional players. All results presented are from inexperienced players (male and female). Although most of the moves were recorded, only 2 hours of recording were taken into account, with slots of 10 minutes for each time of the day (morning, noon and evening) in 4 different days. The tests were divided in 3 main categories: Motion detection, balls detection and trajectories.

For the ***table boundaries detection*** two completely different tables were used, both different from the one used for the prototype (red one) with also different lightning conditions. As boundaries are usually the same colour as the table, there is almost no contrast between them, making it difficult to detect if the light in the surroundings is poor. In well-lit surroundings, by "well-lit" meaning all the usual lights turned on, the table boundaries were always automatically detected (100% of the times) with less than 3 px error that can be corrected with the computer mouse (in the initial configuration menu; see Section 6.6).

***Motion detection*** is the procedure that triggers cue detection or ball detection. In case motion stops, balls should be found. In the 163 moves made, motion detection

has worked 100% of the times, triggering correctly the ball detection algorithm. For the 100% result, the maximum delay obtained between the motion actually stopping and the ball detection triggering was around 2/3 s. This value was due to the fact that $N_p = 40$ previous frames were used. Most users didn't notice this delay, nevertheless, tests were also done using a delay around 1 s (the proposed $N_p = 25$) but in this case motion detection dropped the successful rate to 90% in real exposition conditions.

For the ***ball detection***, in 163 moves made, there were a total of 605 balls to be detected (the application goal is to helping playing pool, with only some of the balls actually placed onto the table for each of the 163 moves). All balls were successfully detected but there were 17 false positives detected. This result was due to some noise that was not filtered correctly. The white ball was also analysed, with it being successfully detected in 354 out of 389 times (91%), with the 19 times not being successfully analysed due to a striped ball being in an area with higher luminosity and its white part facing up and the other 16 times due to the white ball having been potted. Balls detected had maximum of 1cm error of where they were. This error occurred due to some imperfections of the ball detection algorithm, such as noise on the frame acquired by Kinect and distortion of the projector, with the error increasing for furthest balls from the centre of the table (position of the Kinect sensor).

The ***cue detection***, in the above 163 moves, was detected always (100%) as well as the ***strike detection***. Nevertheless, some errors were noticed in the cue detection outside this tests, whenever the players do not obey to the pool rules, e.g., put two cues in the pool table area or there were several people with the hands moving near the table border. For the strike detection no errors were noticed, although a small delay of around 1 s could sometimes be noticed, mostly due to the cue occupying the position of where the white ball previously was.

Finally, the ***shot prediction*** and most problematic test. A test in this category was considered successful, if a player had successfully hit the ball he/she was aiming for, being a bit subjective, since some players don't know how to hold/support the cue

over his/her hand. Taking the above in consideration, predicted trajectories errors only occur due to errors on the ball detections and due to the white ball not being hit in the middle, with all errors being angular, increasing their visual perception for greatest distances. Balls that not bounced were successfully tested in 98% of the cases, while balls that bounced once were successfully tested in 80% and balls that bounced twice were successfully tested in 53% of the cases. More bounces were not included in this test, since their poor results (a bounce considered each touch the white ball does on a table boundary before touching other ball). The trajectories errors of the ball increases as more boundaries are involved, with a higher value for a stronger hits, due to the spin a ball gains when it hits a boundary.

A final note for the ***menus/interface***: all the menus worked as expected, showing no problems. The same was notice with the use of the Leap Motion as the interface. Problems were only detected when the sensor was not correctly used, e.g., when the sensor was not positioned with the green light facing the user.

## 6.8 Conclusions

An application that aids a beginner player to play pool was presented using mainly a Microsoft Kinect 2 sensor, allowing to detect table boundaries, each ball and the cue stick with an excellent accuracy. All the algorithms demonstrated to be very robust against lightning conditions and noise. A projector, placed above the table, shows in real time the computed trajectory line in order to give a player a perception of what is going to happen in that particular move.

The system has two stages: (a) The set-up, which is done the first time that the system is set up (or if the table changes position). In this stage a computer interface helped adjusting all configurable parameters if necessary, which could not be changed later unless repeating this stage. (b) The running stage, where every interaction with the system is done using the Leap Motion sensor and the pool table or screen.

The system works in real time, and all the tests were done in real and not controlled conditions, showing very good results. A comparison with previous systems is difficult, because as for the best of our knowledge there isn't any database or ranking to test this algorithms and, plus, this is the only system working in real time and in real conditions, whereas systems like Denman et al. (2003); Höferlin et al. (2010); Guo and Mac Namee (2007); Shen and Wu (2010); Legg et al. (2011); Ling et al. (2012); Parry et al. (2011) work on video (or video streams) taken from championship of pool or snooker, with very stable and controlled conditions (light, player positions, etc.) during the entire match, and Archibald et al. (2010); Dussault et al. (2009); Landry et al. (2011); Nierhoff et al. (2011) work in a more or less controlled environment because of the robots.

In relation to the four most similar Alves et al. (2013); ARPool (2014); Shih (2014); Shih et al. (2012), there is no data/publication available to do any type of comparison for the ARPool (2014), while comparing it with Alves et al. (2013) (our previous work) the system improved in terms of reliability around 10%-20% (depending on the different tests) relative to the version with the RGB cam, while improving reliability 1%-5% using two Kinect 1 (not published). In a deeper look to Alves et al. (2013), the algorithms were improved in order to detect balls when they were in contact with each other as well as improvement on the precision of detection of all game elements, with also enhancing cue detection when a player has his/her hand near the white ball to play (although it may increase the error of determining the cue direction).

Finally, Shih et al. (2012); Shih (2014), presented a very interesting work in terms of physics of the game, nevertheless they use a very small table in a controlled condition. They use an RGB camera to extract the balls and cue, when applied in real situation, e.g., different lights (pubs, expositions), table cloths, etc. it is expected not to be very reliable in comparison with the present work (see our previous work Alves et al. (2013), and the discussion above). In terms of augmented reality, they only show their output on a computer screen, which is not practical for a player that, besides

being focused on his move, needs also to focus on the computer screen. Comparing their work with the presented in this Chapter is quite difficult, since they have different goals, for sure Shih et al. (2012); Shih (2014) has a better physics than ours, but with different goals to the present system where the main goal is to teach inexperience players how to play. In the other hand, by using the Kinect sensor, system turned to be very reliable and proved to work in any environment using any table cloth, balls or cues.

As mentioned, with the system very stable, near future work will focus on increasing the number of menu options, and improving the augmented reality menu. This includes animations to project onto the table (requested by some users), detection of where the balls were potted and respective colour to implement an automatic score system (requested by the more professional users) allowing to build more statistics. All the physics can be improved if strike force is estimated, as well as the exact position the cue hits the white ball. This last point will be for sure a very changeling goal.

# 7

# Conclusions

In this thesis was presented interaction based intelligent systems using on that regard Microsoft Kinect 3D depth sensor. Three different types of applications were built and shown: art installation, commercial interactive installation and applied to physical games, counting a total of five written papers, where these sensors proved to have very good results when working with robust algorithms in real time.

All the conclusions and respective discussions were done in each Chapter. Although each conclusions are not duplicate, a final remarks are done below, concluding all the work done.

For both art and commercial interactive installations, interacting with users using gestures could be done by manipulating users joints data into two lists, providing

analysis to users historical record, used to understand and recognize users' gestures. For the commercial installation, these lists also proved to be useful when building statistics about users joints or global position or counting the time spent by them while interacting, saving all processed data into a database for later analysis and building heat maps.

Since Kinect doesn't offer a wide-angle of interaction, a method was built to enable more sensors working together, creating a wider area where a user can interact. This method proved to be robust, keeping track of users even when changing between two sensor areas, detecting and recognizing this user as the same and keeping track of him/her and building the same statistical data as above.

For the physical games, a 3D depth sensor was applied to predict snooker or billiard moves. In this subject, the 3D depth sensor proved to have better results than a Full HD web-cam, greatly improving algorithms robustness due to depth capabilities of this sensors, with it's main advantage being the robustness to lightning conditions as well as the distance map returned by this device.

## 7.1 Future Work

Most of the work done and presented in all the Chapters contained new and enhanced algorithms when comparing to their previous Chapter. Although this is true, in the case of the interactive art installation, improvements could be done by adding voice and speech functionalities to the storyboard, triggering new actions on the installation. A Pepper's Ghost real sized installation should also be worked on, since a real sized installation has a greater impact on spectators, allowing the advantages of both holographic and size. On this last point, an alternative installation might be considered using a real sized mannequin with a projector, mapping the video character to this mannequin to create a new reaction on the spectator.

For the commercial installation, the occlusion detection should be worked on. When

an occlusion occurs, the current algorithm searches for new users in the same area a user was lost but instead, this algorithm should use the physical positions stored on the list to determine and extrapolate a new position, searching there instead. Improvements on the head estimation algorithm is also needed, since neither it gives the real position of where the user is looking to but it also has distances issues. To improve it, a Kinect v2 should be used to have a better colour resolution to successfully detect the eyes and attenuate these distance issues. Finally, the skeleton segments descriptor needs improvements to increase the successful rate of users detected by different Kinects. For this, descriptors can also contain information about patterns, turning it more robust to lightning conditions.

For the real time snooker tool improvements should be made on several subjects. Distinguishing and recognizing all balls, instead of the white ball, would be a great improvement to the framework. Balls should also be tracked after players' strike to determine if the ball has been potted or not. With this last point implemented, special effects like balls leaving a trail of fire or ice being cracked can be implemented, also allowing an automatic scoring system. Artificial intelligence could also be used to aid the player making a good move, showing onto the table lines of what he/she should do.

## 7.2 Publications

In the three years of my masters degree, several works were published, with a total of six articles for international conferences, two submitted for journal and other accepted for a Book:

- Rodrigues, J.M.F., **Alves, R.**, Sousa, L., Negrier, A., Cardoso, P.J.S, Monteiro, J., Felisberto, P., Figueiredo, M.J.G., Mendes da Silva, B., Gomes, M., Bica, P. (2016) ***PRHOLO: 360º Interactive Public Relations***, accepted for Handbook of Research on Human-Computer Interfaces, Developments, and Applications, IGI

Global

- Sousa, L., **Alves, R.**, Rodrigues, J.M.F. (2015) ***Augmented reality system to assist inexperienced pool players***, accepted for The Visual Computer.

- **Alves, R.**, Sousa, L., Negrier, A., Monteiro, J., Cardoso, P.J.S., Felisberto, P., Bica, Rodrigues, J.M.F. (2015) ***Interactive 360º Holographic Installation***, accepted for The Visual Computer.

- **Alves, R.**, Negrier, A., Sousa, L., Rodrigues, J.M.F., Felisberto, P., Gomes, M., Bica, P. (2015) ***Interactive 180º Rear Projection Public Relations***, In Proc. Int. Conf. on Computational Science, Reykjavík, Iceland, 1-3 June, pp. 592-601. doi:10.1016/j.procs.2015.05.327

- **Alves, R.**, Sousa, L., Negrier, A., Rodrigues, J.M.F., Cardoso, P.J.S., Monteiro, J., Gomes, M., Bica, P. (2015) ***PRHOLO: Interactive Holographic Public Relations***, In Proc. 3rd Int. Conf. on Advances in Computing, Communication and Information Technology, Birmingham, UK, 26-27 April, pp. 124-128. doi: 10.15224/978-1-63248-061-3-74

- Figueiredo, M.J.G., Sousa, L., Cardoso, P.J.S., Rodrigues, J.M.F., Gonçalves, C., **Alves, R.** (2014) ***Learning Technical Drawing with Augmented Reality and Holograms***, In Proc. 13th Int. Conf. on Education and Educational Technology (EDU '14), Lisbon, Portugal, Oct. 30-Nov. 1, pp. 11-20. ISBN: 978-960-474-395-7

- Sousa, L., Rodrigues, J.M.F., Monteiro, J., Cardoso, P.J.S., Semião, J., **Alves, R.** (2014) ***A 3D Gesture Recognition Interface for Energy Monitoring and Control Applications***, In Proc. 13th Int. Conf. on Applications of Computer Engineering (ACE '14), Lisbon, Portugal, Oct. 30-Nov. 1, pp. 62-71. ISBN: 978-619-7105-30-08

- **Alves, R.**, Madeira, M., Ferrer, J., Costa, S., Lopes, D., Mendes da Silva, B., Sousa, L., Martins, J., Rodrigues, J.M.F. (2014) ***Fátima revisited: An interactive installa-***

***tion***, In Proc. Int. Multidisciplinary Scientific Conf. on Social Sciences and Arts, Varna, Bulgaria, 2-9 Set., pp. 141-148. ISBN: 978-960-474-393-3

- **Alves, R.**, Sousa, L., Rodrigues, J.M.F. (2013) ***PoolLiveAid: Augmented reality pool table to assist inexperienced players***, In Proc. 21st Int. Conf. on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 24-27 June, pp. 184-193. ISBN: 978-80-86943-75-6

# References


Active8-3D (2015). 3D holographic projection displays. http://www.activ8-3d.co.uk. Retrieved: June 30, 2015.

Alves, R., Madeira, M., Ferrer, J., Costa, S., Lopes, D., da Silva, B. M., Sousa, L., Martins, J., and Rodrigues, J. (2014). Fátima revisited: an interactive installation. In *Int. Multidisciplinary Scientific Conf. on Social Sciences and Arts*, pages 141–148. SGEM.

Alves, R., Negrier, A., Sousa, L., Rodrigues, J. M., Felisberto, P., Gomes, M., and Bica, P. (2015a). Interactive 180 rear projection public relations. *Procedia Computer Science*, 51:592–601.

Alves, R., Sousa, L., Negrier, A., Rodrigues, J.M.F., C. P., Monteiro, J., Gomes, M., and Bica, P. (2015b). Prholo: Interactive holographic public relations. In *In Proc. 3rd Int. Conf. on Advances in Computing, Communication and Information Technology, Birmingham, UK*, pages 124–128.

Alves, R., Sousa, L., and Rodrigues, J. (2013). PoolLiveAid: Augmented reality pool table to assist inexperienced players. In *Proc. of 21st Int. Conf. on Computer Graphics, Visualization and Computer Vision*, pages 184–193. Václav Skala-UNION Agency.

Antonio, S., Herrera, R., and Enriquez, E. (2013). Projection's panel of models for touch screen. *Int. J. of Innovative Research in Computer and Communication Engineering*, 1(9):2057–2064.

Archibald, C., Altman, A., Greenspan, M., and Shoham, Y. (2010). Computational pool: A new challenge for game theory pragmatics. *AI Magazine*, 31(4):33–41.

ARPool (2014). ARpool, augmented reality: Pool. http://rcvlab.ece.queensu.ca/~qridb/ARPOOL.html. Retrieved: Set. 3, 2015.

AssociaçãoDevotos (2014). Primeira aparição de Nossa Senhora de Fátima. http://www.devotosdefatima.org.br/aparicoes2.html. Retrieved: May 16, 2014.

Asus (2014). Asus Xtion Pro. http://goo.gl/HxQcli. Retrieved: Nov. 10, 2014.

Cardoso, P., Rodrigues, J., Carlos, L., Mazayev, A., Ey, E., Corrêa, T., and Saleiro, M. (2015). A freehand system for the management of orders picking and loading of vehicles. In *Accepted for 9th Int. Conf. on Universal Access in Human-Computer Interaction, Los Angeles, CA, USA*.

Chiang, I.-T., Tsai, J.-C., and Chen, S.-T. (2012). Using Xbox 360 Kinect games on enhancing visual performance skills on institutionalized older adults with wheelchairs. In *IEEE Int. Conf. on Digital Game and Intelligent Toy Enhanced Learning*, pages 263–267.

Cruz, L., Lucio, D., and Velho, L. (2012). Kinect and RGBD images: Challenges and applications. In *IEEE Conf. on Graphics, Patterns and Images Tutorials*, pages 36–49.

Deleuze, G., editor (1990). *A Imagem-Tempo*. São Paulo: Brasiliense.

Denman, H., Rea, N., and Kokaram, A. (2003). Content-based analysis for video from snooker broadcasts. *Computer Vision and Image Understanding*, 92(2):176–195.

Doyle, J. and Jun, F. (2013). GestureCloud: gesture, surplus value and collaborative art exchange.

D'Strict (2014). 3D sensing holographic installation. http://goo.gl/CyqKXe (retrieved 26/2/2014).

Dussault, J., Greenspan, M., Landry, J., Leckie, W., Godard, M., and Lam, J. (2009). Computational and robotic pool. *Chap. XII of Digital Sport for Performance Enhancement and Competitive Evolution: Intelligent Gaming Technologies, IGI Global*, pages 194–209.

El-laithy, R. A., Huang, J., and Yeh, M. (2012). Study on the use of Microsoft Kinect for robotics applications. In *IEEE/ION Position Location and Nav. Symp.*, pages 1280–1288.

Figueiredo, M., Sousa, L., Cardoso, P., Rodrigues, J., Goncalves, C., and Alves, R. (2014a). Learning technical drawing with augmented reality and holograms. In *Int. Conf. on Education and Educational Technology*, pages 11–20.

Figueiredo, M. J., Cardoso, P. J., Goncalves, C. D., and Rodrigues, J. M. (2014b). Augmented reality and holograms for the visualization of mechanical engineering parts. In *Int. Conf. on Information Visualisation*, pages 368–373. IEEE.

Flyway (2015a). 3D holographic projection – the future of advertising? `http://flyawaysimulation.com/news/3630/3d-holographic-projection-future-of-advertising`. Retrieved: June 30, 2015.

Flyway (2015b). AVA - advanced virtual assistant. `http://airportone.com/airportvirtualassistancesystem.htm`. Retrieved: June 30, 2015.

Fong, S., Zhuang, Y., Fister, I., and Fister Jr, I. (2013). A biometric authentication model using hand gesture images. *Biomedical engineering online*, 12(1):111.

Gabdulkhakova, A. and Kropatsch, W. G. (2014). Video analysis of a snooker footage based on a kinematic model. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 223–232. Springer.

Godbehere, A. B., Matsukawa, A., and Goldberg, K. (2012). Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *American Control Conf. (ACC), 2012*, pages 4305–4312. IEEE.

Guo, H. and Mac Namee, B. (2007). Using computer vision to create a 3D representation of a snooker table for televised competition broadcasting. *The 18th Irish Conf. on Artifical Intelligence and Cognitive Science*.

Höferlin, M., Grundy, E., Borgo, R., Weiskopf, D., Chen, M., Griffiths, I. W., and Griffiths, W. (2010). Video visualization for snooker skill training. In *Computer Graphics Forum*, volume 29, pages 1053–1062. Wiley Online Library.

Holus (2015). Holus: The interactive tabletop holographic display. `http://www.digitaltrends.com/cool-tech/holus-tabletop-hologram-kickstarter`. Retrieved: June 30, 2015.

ImmaculateHeart (2014). Immaculate heart publications: Fatima! through the eyes of a child. `http://www.fatima.org/exclusives/fatimachild2.pdf`. Retrieved: May 16, 2014.

Jiang, R., Parry, M. L., Legg, P. A., Chung, D. H., and Griffiths, I. W. (2013). Automated 3-D animation from snooker videos with information-theoretical optimization. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(4):337–345.

Kamizono, T., Abe, H., Baba, K., Takano, S., and Murakami, K. (2014). Towards activity recognition of learners by Kinect. In *IEEE 3rd Int. Conf. on Advanced Applied Informatics*, pages 177–180.

Kinect (2014a). Kinect 2 for Windows. `http://www.microsoft.com/en-us/kinectforwindows/`. Retrieved: May. 03, 2015.

Kinect (2014b). Kinect for Windows. `http://goo.gl/fGZT8X`. Retrieved: Nov. 10, 2014.

Kondori, F., Yousefi, S., Liu, L., and Li, H. (2014). Head operated electric wheelchair. In *IEEE Southwest Symp. on Image Analysis and Interpretation*, pages 53–56.

Landry, J.-F., Dussault, J.-P., and Mahey, P. (2011). Billiards: an optimization challenge. In *Proc. of The 4th Int. Conf. on Computer Science and Software Engineering*, pages 129–132. ACM.

Larsen, L. B., Jensen, R. B., Jensen, K. L., and Larsen, S. (2005). Development of an automatic pool trainer. In *Proc. of the 2005 ACM SIGCHI Int. Conf. on Advances in computer entertainment technology*, pages 83–87. ACM.

Leckie, W. and Greenspan, M. (2006). An event-based pool physics simulator. In *Advances in Computer Games*, pages 247–262. Springer.

Lee, H. Y., Kim, J. Y., and Lee, W. H. (2014). Interactive digital art based on user's physical effort with sensor technology. *Int. Journal of Software Engineering & Its Applications*, 8(3):211–216.

Legg, P. A., Parry, M. L., Chung, D. H., Jiang, R. M., Morris, A., Griffiths, I. W., Marshall, D., and Chen, M. (2011). Intelligent filtering by semantic importance for single-view 3D reconstruction from snooker video. In *18th IEEE Int. Conf. on Image Processing (ICIP)*, pages 2385–2388. IEEE.

Ling, Y., Li, S., Xu, P., and Zhou, B. (2012). The detection of multi-objective billiards in snooker game video. In *3rd Int. Conf. on Intelligent Control and Information Processing (ICICIP)*, pages 594–596. IEEE.

Litefast (2013). Litefast MAGIC displays. `http://www.litefast-display.com/products/litefast-products/litefast-magic/litefast-magic.html`. Retrieved: February 28, 2013.

Martins, J. M., Rodrigues, J. M., and Martins, J. C. (2015). Low-cost natural interface based on head movements. *Int. Conf. on Software Development and Technologies for Enhancing, Fraunhofer FIT, Sankt Augustin, Germany*.

Mathew, A., Rogers, Y., and Lloyd, P. (2011). Post-it note art: evaluating public creativity at a user generated art installation. In *Proc. of the 8th ACM Conf. on Creativity and cognition*, pages 61–70. ACM.

Mihaylova, E., editor (2013). *Holography - Basic Principles and Contemporary Applications*. InTech.

Mora, K. and Odobez, J.-M. (2012). Gaze estimation from multimodal Kinect data. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, pages 25–30. IEEE.

Moser, J. M. (2014). Tupac lives! what hologram authors should know about intellectual property law. http://goo.gl/zMxTZP (retrieved 26/2/2014).

Motion, L. (2014). Leap Motion. `https://www.leapmotion.com/`. Retrieved: Nov. 10, 2014.

Nierhoff, T., Kourakos, O., and Hirche, S. (2011). Playing pool with a dual-armed robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3445–3446. IEEE.

Parry, M. L., Legg, P. A., Chung, D. H., Griffiths, I. W., and Chen, M. (2011). Hierarchical event selection for video storyboards with a case study on snooker video visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1747–1756.

ProDisplay (2014). ProDisplay. `http://www.prodisplay.com/download/rear-projection-spec.pdf`. Retrieved: May 16, 2014.

Rahman, M., Poon, B., and Ashraful Amin, H. Y. (2015). Support system using Microsoft Kinect and mobile phone for daily activity of visually impaired. In *Transactions on Engineering Technologies*, pages 425–440. Springer.

Rearpro (2015). Paradigm audio visual. `http://www.rearpro.com`. Retrieved: June 30, 2015.

Rennie, J. (2014). The Tupac hologram, virtual ebert, and digital immortality. http://goo.gl/fQYUt2 (retrieved 26/2/2014).

Russ, J. C. (2010). *The image processing handbook*. CRC press.

SantuárioFátima (2014). Santuário de Fátima. `http://www.santuario-fatima.pt/portal/index.php?lang=EN`. Retrieved: May 16, 2014.

Shen, W. and Wu, L. (2010). A method of billiard objects detection based on snooker game video. In *2nd Int. Conf. on Future Computer and Communication*, volume 2, pages V2–251. IEEE.

Shih, C. (2014). Analyzing and comparing shot planning strategies and their effects on the performance of an augment reality based billiard training system. *Int. Journal of Information Technology & Decision Making*, 13(03):521–565.

Shih, C., Koong, C.-S., and Hsiung, P.-A. (2012). Billiard combat modeling and simulation based on optimal cue placement control and strategic planning. *Journal of Intelligent & Robotic Systems*, 67(1):25–41.

Sousa, L., Rodrigues, J., Monteiro, J., Cardoso, P. J., Semião, J., and Alves, R. (2014). A 3D gesture recognition interface for energy monitoring and control applications. pages 62–71.

Sprott, J. C. (2006). *Physics Demonstrations: A sourcebook for teachers of physics*. Univ of Wisconsin Press.

Structure (2014). Structure Sensor. `http://structure.io/`. Retrieved: Nov. 10, 2014.

Suzuki, S. et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46.

Unity (2014). Unity 3D. `https://unity3d.com/pt`. Retrieved: Nov. 10, 2014.

Vizoo (2015). Cheoptics. `http://www.vizoo.com/flash`. Retrieved: June 30, 2015.

Wei, B., Guan, T., Duan, L., Yu, J., and Mao, T. (2015). Wide area localization and tracking on camera phones for mobile augmented reality systems. *Multimedia Systems*, 21(4):381–399.

Weiss Cohen, M., Frid, A., Malin, M., and Ladijenski, V. (2015). Generating 3D CAD art from human gestures using Kinect depth sensor. *Computer-Aided Design and Applications*, (ahead-of-print):1–9.

Yun, Y., Changrampadi, M. H., and Gu, I. Y. (2014). Head pose classification by multi-class AdaBoost with fusion of RGB and depth images. In *IEEE Int. Conf. on Signal Processing and Integrated Networks*, pages 174–177.