Gestural Interface with a Projected Display Game

by

Gianluca Vatinno

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science (Interactive entertainment technology)

University of Dublin, Trinity College

September 2010

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Gianluca Vatinno

September 10, 2010

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Gianluca Vatinno

September 10, 2010

Acknowledgments

Thanks to my supervisor Dr. Gerard Lacey for his wise and expert guidance in this project. Thanks to Dr. John Dingliana Director of the M.Sc. for his always friendly support during the year. Thank to Jiang Zhou for his helpful advises. A special thanks goes to my class mates, also my friends, who supported me during this M.Sc. adventure. Thanks to Richard O'Riordan for his helping hand with Maths; Thanks to Da Cruz Ramalhao Carlos Jorge for his always highly qualified and present support; Thanks to Marco Conti for is constant daily help and company as desk mate; Thanks to Brendan Carroll for his qualified help in LaTex and language. Most of all thanks for their friendship. Thanks to my family for their omnipresent support.

Gianluca Vatinno

University of Dublin, Trinity College
September 2010

Gestural Interface with a Projected Display Game

Gianluca Vatinno
University of Dublin, Trinity College, 2010

Supervisor: Dr. Gerard Lacey

Today technology is moving fast towards interaction innovation; new way of Human Computer Interactions (HCI) are being researched, especially in the field of Interactive Entertainment Technology. The history of console controllers has drawn the direction for future interaction with interactive applications, which points to a "controllerless" approach. This research follows on that direction. The research project presents a system which uses a Pico-projector to project a game onto a table top and uses hand gestures, recognized by a camera, to interact with game elements. A technology inspection is carried out throughout the research to highlight subtle issues of the coupled technology camera-projector. Methods and techniques which provide the best accuracy in hand segmentation are analyzed by fast prototyping. A final demo shows the research project feasibility, presenting the remaking of the classic Labyrinth Wood Maze in a digital videogame playable with hand gestures. In the end a technical evaluation assesses the Gestural Interface Accuracy, Robustness and Repeatability; A user impact evaluation reports the users' opinions on the overall system.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Today technology is moving fast towards interaction innovation; new way of Human Computer Interactions (HCI) are being researched, especially in the field of Interactive Entertainment Technology. Research is pointing towards mitigating or even eliminating the presence of any form of extraneous and invasive controller from interactive applications. This phenomenon has been undergoing acceleration since just few years ago. In fact looking back at the history of video game consoles is overwhelming that the complexities of video game controllers have been keeping on increasing since the first invented game pad. The Paddle in 1972 (Fig. 1.1a), together with the legendary game Pong, as the first game console controller had only one buttons and a turning wheel; it could appear very straightforward to use, perhaps not at that time. As new consoles have been released the number of buttons on new controllers appeared to increase exponentially; reasons lie in the complexity of new video games which allowed many more possible actions; moreover the competition with the PC game industry helped console controller complexity to grow up; Pc games users in fact have a very rich controller full of keys: the keyboard. Other controllers in order of release were:

Sega Genesis Controller (eight buttons - Fig. 1.1b), Playstation controllers (ten buttons - Fig. 1.1c), Nintendo 64 (eleven buttons - Fig. 1.1d), Microsoft Xbox (fourteen buttons plus two thumbsticks - Fig. 1.1e). These controllers made happy growing generations of gamers, but they have restricted video game audience to pure technology background individuals; casual players with no technology background have no time or patient to learn how to use these tricky controllers especially in casual interactions. Nintendo releasing its Wii console (Fig. 1.1f) has drawn a turning point allowing play-

(a) Paddle - 1972 (b) Sega Genesis - 1989 (c) Sony Playstation - 1994 (d) Nintendo 64 - 1996

(e) Microsoft xbox - 2001 (f) Nintendo Wii - 2006 (g) Natal - being released

Figure 1.1: Video game console controllers history

ers interacting with video games using natural movements through wireless handy little controllers. Noteworthy this year Microsoft is going to release a brand new technology called Project Natal [6] (Fig. 1.1g) which will consent players to interact with a console having no controllers, making the player itself a controller. These two latter technology attract a larger audience, with no computer science background, to buy and use them. Attracting more users to this new technology has positive economical consequences in companies business and profits. The history of console controllers (Fig. 1.2) has drawn the direction for future interaction with interactive applications, which points to a "controllerless" approach.

All the cited technology, have something in common: they allow players to interact with consoles connected to displays; all of them, even with different interaction paradigms of interaction, need displays that which are not mobile devices.

What if displays were replaced by projectors and what if projectors were so tiny to be embedded in mobile phones? Mobile phones companies are developing Pico-projectors [3] which result to be tiny mini projectors. Some researchers have proposed novel interfaces for this new capability. Designing a new interaction paradigm which exploits a camera to recognize natural hand gestures and a projector to project an interactive application in place of a display would represent an innovative interactive approach to multimedia applications. This research project will present a system which uses a Pico-projector to project a game onto a table top and uses hand gestures, recognized by a camera, to interact with game elements.

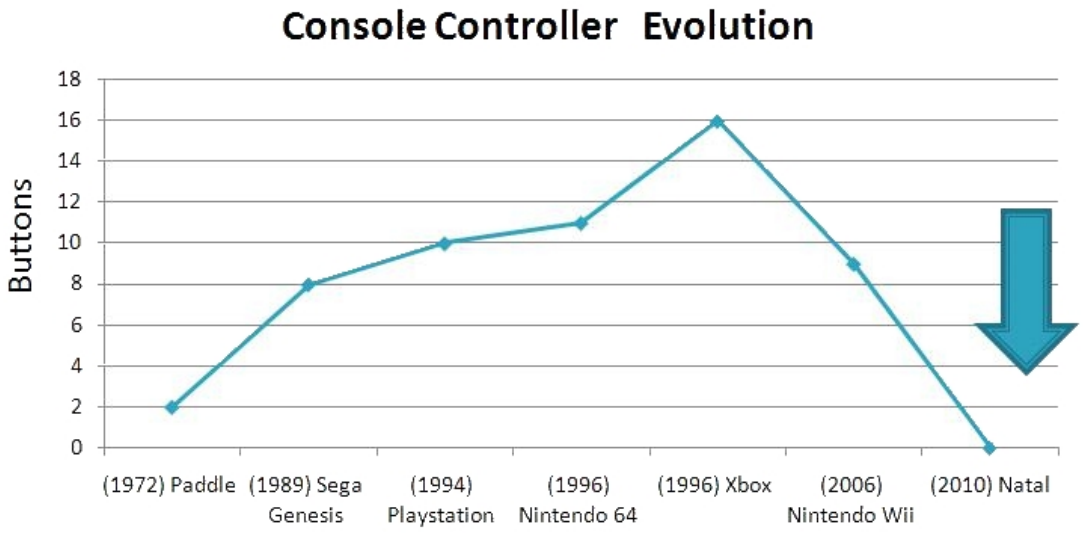


Figure 1.2: Console Controller Evolution

1.2 Motivations and aims

Allowing players using their hands as natural controllers, exploring a new market of casual games which does not require users to own a technical or computer science background, removing the constraint of playing a game on a display are all motivations of this research project.

As mentioned in the introduction today video games are more complex and controllers available are even more difficult to use for not assiduous gamers; people nowadays have less time to dedicate to games but they may have few minutes to play a casual game; besides if the time to understand a controller is drastically reduced, more users with no technology background will became a potential new consumers along with who is already in the game market. Current technology is invasive but if it is perceived easy and natural people will accept it faster. Moreover images on classic displays are constrained in size and shape by physical frames; projectors give to the users more freedom in term of mobility and image size and consequently in interactive applications; indeed projectors allow images to be scalable and visible on different surfaces. In the end porting the propose interaction on mobile phone equipped with Pico-projector and camera would represent a new form of pocket entertainment resulting not constrained by mobile phone display sizes and their physical locations.

The following research is aimed at studying, designing and developing a gestural interface to interact with a projected display game. The project will exploit a tiny Pico-projector to project a game on a planar surface (table top). A camera will recognize human natural hand gestures exploited to command the game elements. Furthermore the project will present a game playable from one or two players at the same time. Main goals of this research are:

- Investigating the technology camera-projector through application prototypes highlighting innovative aspects, limitations and any issue may arise during the

investigation.

- Designing and developing a gestural interface able to recognize simple hand gestures used as input for a game.
- Designing and developing a table top video game for one or two players.
- Evaluating the gestural interface implemented through measurements of his accuracy and robustness.
- Evaluating the overall video game produced through users trials with support of user questionnaires.

Chapter 2

Background

Literature presenting gestural interfaces interacting with a table top projected display game is quite sporadic, although a wide range of studies of table top display applications with hands and fingers interaction is widely present. Prototypes of multi-touch tabletop technology which recognize touches on displays by internal reflection of light [14] and electrical signals [19] have been already studied several years ago. Commercial examples of these applications are Microsoft surface [4], Perceptive Pixel [5] and by far the most popular multi-touch product on mobile platform: the Apple Iphone [1]; all of them allow users to manipulate interfaces exploiting touch natural hand and finger gestures.

Gesture recognition to interact with a projected interactive application is treated by Mistry's and Maes's work [25], presented earlier as WUW (Wear Ur World) [26]; it presented a computer vision gestural interface able to recognize natural hand gesture through a camera; the hardware was composed of a tiny projector and a camera coupled in a pendant like a mobile wearable device; gesture recognition was obtained by visual tracking of fiducial markers (coloured markers); fiducial marker detection is a solid techniques but is limited in the number of unique fiducial trackers with multi user interaction; besides a mirror was required to reflect the projected image in front of the user; the use of a mirror could be substitute using a software warping functions on the

projected images within a certain angle of projection; adapting projected images to different surfaces is fully treated by Bimber in his Lecture notes [8]. It applies to many field of augmented reality such as virtual motion, large environments recreation, curved shape Displays images and holographic. A wearable mobile projector-camera system (called Interactive Dirt [24]) was applied in a military context to improve and maintain situational awareness; it resulted to reduce situational impairment effect for using of hand-held device which in a military situation is of vital importance; a projector was used to create a interface on different surfaces and a Nintendo Wii mote Infrared camera employed to detect pointing fingers or pointing infrared sticks representing gestures to command the projected application; the described approach appeared solid in a hostile environment despite light condition or complex background environment thanks to infrared recognition; equipment revealed itself highly invasive as a wearable ubiquitous technology though. Molyneaux work's [27] presents a projector-camera system in order to enable interaction with smart tangible object; although a simple touching recognition on a button lying on the projected image has been implemented through template matching recognition which worked out robustly for simple one finger touching interactions.

Key topic in all previous cited works is the hand and gesture recognition. A deeper examination on the literature related to this topic shows that a considerable amount of work has been carried out in this field. Nikola, Ribari and Grad [28] studied hand palm recognition and analyzed different techniques for biometric hand recognition; noteworthy was their approach to recognize fingers and palm from a palm grey-scale image on which segmentation was applied followed by computation of geometrical parameters with recognition of nine rectangular interesting areas, one on the palm and two on each finger; the solution revealed solid results for still frames of hand palms, but it results not suitable for continuous online hand recognitions. Cerlinca's paper [11] presented a faster posture recognition through head recognition and skin recognition filter to detect

hands; posture was detected exploiting Leonardo da Vinci's theory which states that the size of human body is a function of the size of their head sizes. An earlier approach to hand and finger recognition was proposed by Malik and Laszlo [23] which presented a hand blob recognition after a background subtraction; fingertips were detected by finding strong peaks along the blob perimeters; this approaches represents a valid solution but suffer in solidity when the background increase in complexity. The project Wii made by Jonny Lee [2] shown how it is possible to track fingers immersed in a infrared LED array exploiting a Wiimote infrared camera capturing light reflections produced on the fingers. This method is very robust but needs a number of technology equipment to be considered intrusive to a natural hand interaction. An important contribute to the literature is presented by the Alexandra Stefan's approach [32] in recognizing translation and scale invariance gestures; the core of their technique was based of skin recognition and motion detection of feature within the skin parts recognized; motion of feature was already invariant to translation, the scale invariant detection was obtained by a face detection and a coordinate transformation of detected hands in head centered coordinate; the algorithm result valid but it required a training phase with coloured gloves. An outstanding result in hand fingertips recognition has been carried out by Byungsung and Chun [18]; hand shapes were recognized by skin color base segmentation; fingertips were identified by investigating the contour curvatures from the segmented hand shapes; information from camera calibration were also exploited to generate a hand coordinate system which coupled with optical flow produced position and orientation information in space; three kinds of gestures (rotation, scaling and translation of virtual objects) were detected; a template matching with hand posture patterns was applied as hand gesture recognition. A novel hand detection approach is presented by Choi [16]. The proposed method aimed to detect hand gestures in a natural manner and immerse cluttered backgrounds hands and gestures on the assumption that a hand fore-forearm region has different brightness from other skin-colored regions.

A generalized statistical model is used to detect skin regions, Principal Component Analysis and neural network to recognize gestures. Pattern recognition methods using Support Vector Machine (SVM) applied to gesture recognition is treated by Yun [33] and Yen [12]. The former presented hand detection thought Viola Jones method and uses Support Vector Machine pattern recognition trained with Hu invariants moments of hand images. The method shows hight rate of hand gesture detection performed in a reasonable time frame. The latter paper presented also a SVM pattern recognition method except for a pre-processing step of the sample training images; in fact hand images, after being converted in greyscale, they underwent an histogram equalization process which improved their contrast and consequently the overall results.

A common techniques often found in reading the cited works is represented by the Skin detection applied to hand detection. The topic is treated by Elgammal [7] where primers of skin detection are explained. A comparison of skin detection results with different colour spaces is presented by Phung [31]. These last papers highlight that static Skin detection models result affected by false positives and they appear to be very dependent by lighting conditions. An Adaptive Skin Detection was proposed by Farhad Dadgostar [13]. This algorithm can produce better skin detection that static skin model algorithms; it analyze the Hue histogram in HSV colour space of input image maximizing the accuracy of the Hue threshold based on motion detection techniques; the motion information are used to refine the Hue thresholding according to the assumption that candidate skin pixels are more likely to be skin ones if they are moving. A novel energy based blob analysis for improving precision of skin segmentation is presented by Kawulok [17]; the blob analysis algorithm reduces false positives and improves skin segmentation precision; pixels colour from a face detected area are used to adapt the skin model improving the final results.

Chapter 3

Design

The background research has shown how this research work may be placed within the innovative gestural interfaces and novel visualization technique research applied to the entertainment industry. The core research work of this paper (Fig. 3.1) can be divided in two main conceptual research areas; the first aimed to design and develop an innovative, simple but effective way to interact with entertainment applications; the second is aimed at designing and developing an entertainment application which will effectively meet the interaction designed. Collateral, but not secondary, research implies the camera-projector technology investigation which may take place at any stage of the development. An overall final evaluation will present technical results for the presented gestural interface; user trial tests will bring an audience opinion and evaluation to the final demo and its impact in the video game world.

3.1 The plan

The entire development process is divided in three phases:

- Design.
- Rapid prototype.

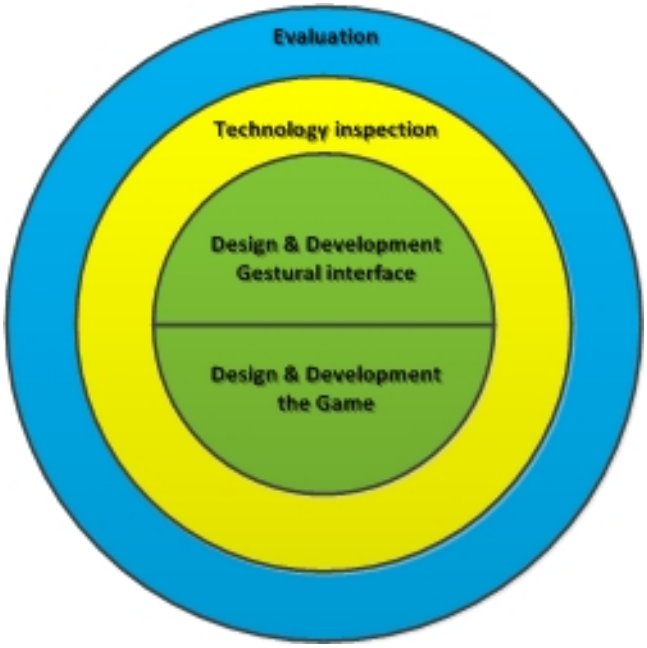


Figure 3.1: Conceptual research work subdivision.

- Final Release.

The design part is the one being treated as follow. A rapid prototype to the solution is highly useful to inspection the technology camera-projector. The choice of a rapid prototype lies in the lack of implicit knowledge about the camera-projector technology and its related issues. The prototype small gestural interface with a small simple game would highlight subtle issues in order to address them in the final release. Moreover a prototype helps to evaluate the designed approaches and methodologies adopted for the solution; an overall technical prototype evaluation can suggest potential changes in the design that otherwise can appear too late in the development process of the final release. The final release addresses all the issues raised in the design part and the prototype phase using the best approaches resulted from their analysis and practical evaluations.

An Agile development process [30] has been adopted for this research work; macro problems has been subdivided in tasks archived in weekly sprints.

3.2 Divide and Conquer

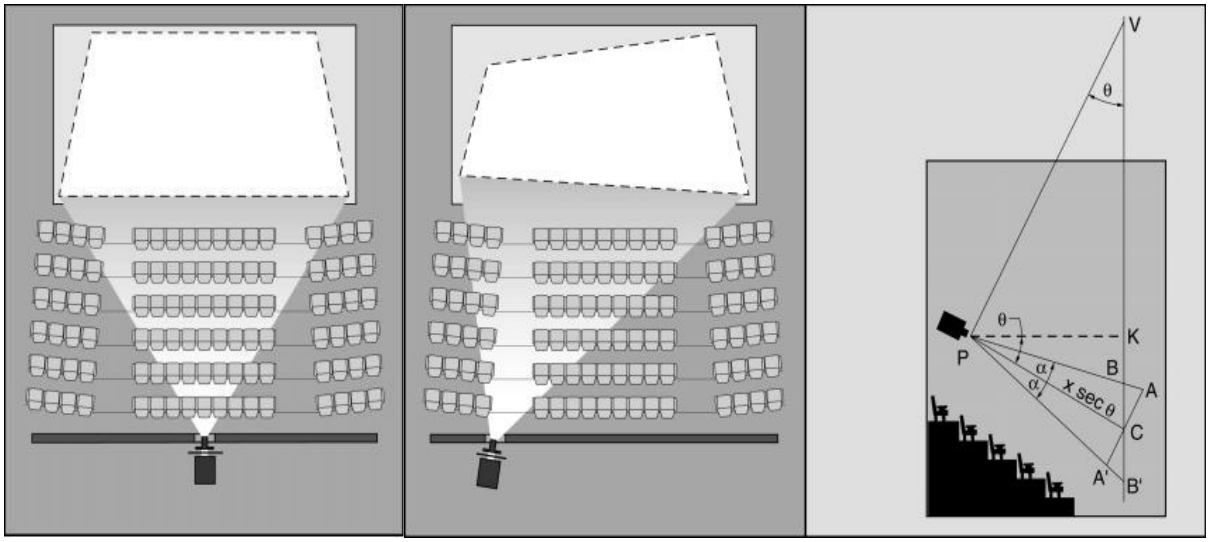
A preliminary analysis on how to design a video game playable with hand gestures suggests subdividing the main problem in two main areas:

- Gestural Interface Design
- Game Design.

Since the aim of this research is to recognize hand gestures through a camera the former area involves a branch of computer science proper called Computer Vision. This branch studies methods and algorithms to extract different kind of information from two dimensional images; often the image processing is exploited to build real time application involving somehow Computer Vision. The latter area instead involves cutting edge video games development techniques covering different topics such as: Gameplay, Physics Simulation, Artificial Intelligence, Sounds and etc. Techniques and methodologies useful to tackle the design of a gestural interface and the actual game design are disclosed as follow.

3.3 Hand Gesture Interface Design

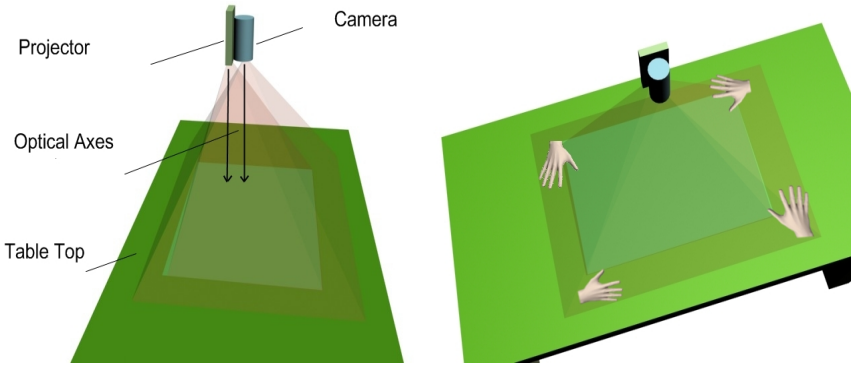
As any Computer Vision application requires, the gestural interface being described also requires a set of assumptions and requirements to be defined before seeking the best approach to solve the problem. In Computer Vision there are plenty algorithms to adopt for the solution of a similar problem; only defining and constraining the environment in which the application works, a more accurate research and discussion about what techniques meet the given problem requirements is possible. The first and most important assumption is that the camera during the entire application is static; the camera does not change its position during the user interaction. This is a very important assumption as the attempt to recognize moving hands would be harder or



(a) Vertical Keystone Distortion (b) Horizontal and Vertical Keystone Distortion (c) Projector Geometry.

Figure 3.2: Keystone Effect. *Copyright Petrozzo, Singer* [29]

even vain with a moving camera. Useful to define, at this particular stage, is what the camera sees during the interaction. The camera at any time of the process should be able to see the entire image projected on a flat surface, in this case a table top. The camera, being able to see the entire projected image, allows the application to detect it capturing information about its shape and dimensions. Consequence of this last assumption is the study of the projector position respect to the table top surface and the camera position respect to the projector itself. The projector position and the angle respect its projecting surface may cause the insurgence of the well studied Keystone effect [29] also well known as Tombstone effect. The Keystone effect is a geometric distortion in the projected image that brings a trapezoidal image of a nominally rectangular image (Fig. 3.2a and 3.2b); it usually occurs when a picture is projected from a position such that the line of sight or optical axis of the projector is not precisely orthogonal to the screen or projecting surface (Fig. 3.2c). To avoid keystone issue the projector position has to be as orthogonal as possible to the table top



(a) Camera projector setting (b) Hand gestures interaction

Figure 3.3: Camera projector setting.

surface. Consequently to allow the camera to see the entire projected image without considerable distortions, it has been decided to couple the projector and the camera together having both lenses to be as coaxial as possible (Fig. 3.3a).

The first step toward the hand gesture detection is to identify where in the camera captured frames the hands are suppose to be found. This topic will be elaborate in more details on the Prototype Chapter; indeed its technical implications will be studied at that stage. At this stage the potential interaction of users' hands in the camera frames can be assumed along the projected image sides (Fig. 3.3b). Having set assumptions for the Gestural interface design, a more accurate overview of potential techniques for hand detection can be treated. Interesting Computer Vision techniques that might suit the aim of detecting hands are: Skin Detection, Motion Detection and Background Subtraction; these three techniques used with proper assumption can help in detecting hand blobs. The following section will treat them in more details; in the prototype and implementation chapters they will be implemented and evaluated in order to choice the ones that provide the best accuracy in the Hand Detection.

3.3.1 Skin Detection

Skin detection is the process of finding skin-coloured pixels and regions in a digital image or a video. This process is typically used as a pre-processing step to find regions that potentially have human faces and limbs in images. [7]. One of the biggest challenges in skin detection is that skin colour in an image depends on the illumination conditions where the image was captured. Representing skin colour as invariant or at least insensitive to lighting changes is an important challenge. Skin color occupies a part of a color space (in any color space) called skin color cluster. Many algorithms have been proposed for skin color pixel classification such as static classifier, Bayesian classifier, Gaussian classifiers and many more complex ones. In a skin classifier an image pixel is classified and labelled whether it is skin or non-skin one, given a model of the skin color cluster in a given color space. Identifying a pixel in a captured image as $p(i, j)$, a skin classifier will assign a label to it as (eq. 3.1).

$$p(i,j) = \begin{cases} 0 & \text{If classified as non skin} \\ 1 & \text{If classified as skin} \end{cases} \tag{3.1}$$

The simplest static classifier proposes a set of fixed skin thresholds on the image pixel chrominance; although it results quite poor in accuracy; skin in fact introduces colour bias depending on human race and it changes colours under different illumination conditions. Therefore, skin detection methods that use static skin colour models result not robust to the changeable skin nature. Skin classification can be defined using probability; given a pixel with colour c and $P(skin|c)$ as the probability of it being skin pixel. Once this probability is computed, the pixel is labelled as a skin pixel if the probability is greater than a threshold value and non-skin otherwise. The probability

of $P(skin|c)$ is calculated using Bayes rule (eq. 3.2).

$$P(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + P(c|notskin)P(notskin)} \tag{3.2}$$

In the Bayes rule $P(c|skin)$ defines the posterior probability of a pixel being skin given its colour. P(skin) is a prior probability of a random pixel being a skin without knowing its colour. $P(c|notskin)$ is the posterior probability that a pixel given its colour is not a skin pixel; $P(notskin)$ is the probability of a pixel not to be a skin one without information about its colour. $P(c|notskin)$ and $P(c|skin)$ probability class need both to be modelled.

Skin detection suffers of limitations and issues. Skin colour is affected by ambient light which is unknown in many situations; moreover different cameras produce different colours and even the same subject, under the same lighting conditions produce different results; skin colours change from person to person and many background objects and materials result to have similar skin colours; For example, wood, leather, skin-coloured clothing, hair, sand, cause any skin detector to have many false detections in the background if the environment is not constrained.

3.3.2 Motion Detection

Motion detection techniques are widely diffuse in tracking applications and optical flow estimations. Motion detection is usually used to estimate the motion between two frames without any prior information about image contents. The result of this operation is an estimation of pixel velocities or displacements from the first frame to the second one. The result image that describes pixel velocities or displacement is commonly called optical flow. One of the biggest challenges in motion detection is detect movement within a homogeneous region of pixels; for example the motion detection of an object with homogeneous colour is possible for its edges but for the

pixels within the shape the motion would be zero because the pixel colours would result constants. In order to solve this problem more accurate algorithms have been proposed such as Lucas-Kanade [20] and Horn-Schunk.

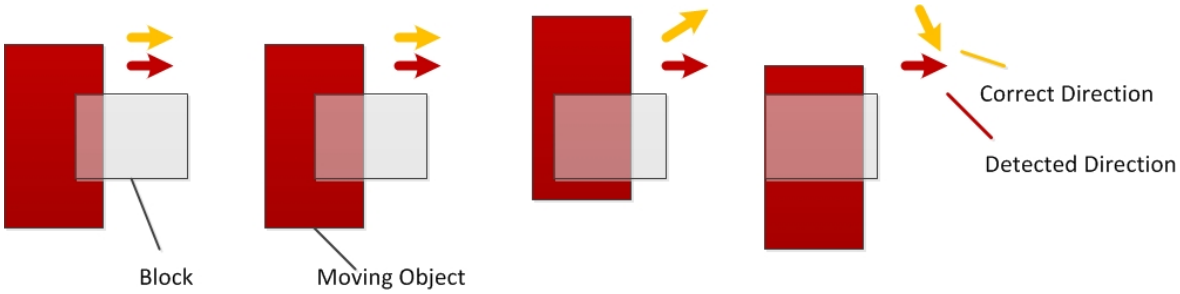


Figure 3.4: Aperture Problem

Lucas-Kanade algorithm is a widely used technique in the computer vision applications. Lukas-Kanade approach instead of estimating motion for single pixels it considers small blocks of pixels estimating the local motion within the blocks. Block size is parametric but using small blocks a large motion cannot be detected. A solution to this issue is a variation of the Lukas-kanade original method called pyramidal Lukas-Kanade; as its name suggests, it creates an image pyramids where the top image presents lower details while the bottom image presents finer details. Tracking the motion over the image pyramids allow large motions to be caught. One of the biggest problem in motion detection techniques is the well know aperture problem (Fig. 3.4). The problem arises when a small block (aperture) is used to detect motion. When the motion is detected, usually edge are visible rather than corners; only visible edges are not sufficient to detect exactly in which direction an object is moving.

3.3.3 Background Subtraction

One of the most used computer vision techniques in contexts where the camera has fixed position is surely the background subtraction. The background subtraction technique is composed of two steps; the first focused on gathering information about the

background creating so its model; the second part consists in the comparison of the background model against the current frame, segmenting the foreground parts. One of the most important assumptions for background subtraction techniques is that the background is considered to be static or that changes over a reasonably long period of time. Background subtraction considers pixels independently when comparing with the current images. Lighting condition changes over time is one of the weaknesses of background subtraction. During the day duration lighting conditions surely vary even if slowly they affect the background accuracy; for these reasons the background model should be re-calculated or updated over time. Morphological operations are also related to background subtraction. Usually after the subtraction of the model from a current image, the resultant foreground image presents noise due to noise in the current image or lighting condition changes or movements in the background. Erosion and dilation operation respectively help the cleanup of the foreground image from the noise and improve the connection of close large pixel area.

3.3.4 Pattern Recognition

One of the most used techniques in the field of hand gesture recognition is surely the Pattern recognition. Pattern recognition is one of the methods used in machine learning to turn data into useful information extracting rules from the data. In computer vision the pattern recognition is used to extract information from images. In order to use pattern recognition with images, data need to be extracted from them. This operation is commonly called *feature extraction*. Pattern recognition algorithms work on two different set of data called respectively *training set* and *test set*. The training set is the set of data created from the feature extraction and the test set is used to test features by the found pattern rule. In Pattern Recognition there are two kind of training set: supervised data and unsupervised one. When the training set is provided to a pattern recognition algorithm with labels (eg. different classes are already assigned to subset

of feature vectors) than the method is called supervised pattern recognition. When instead the training set is provided without any label or classes than the algorithm is called unsupervised pattern recognition; example of unsupervised pattern recognition is Kmean that requires in input an unsupervised training set and the number of clusters (patterns) to separate the training set; for the aims of this research project only supervised pattern recognition algorithms are analyzed. One of the simplest pattern recognition classifier is called K-nearest neighbors (KNN). The training set in KNN algorithm is supervised, therefore class labels are provided along with the features vectors. A new feature vector is classified according to the majority vote of its K nearest feature vectors; in effect KNN algorithm measures the distance between a query feature vector and a set of feature vectors (training set). Different distance function between two feature vectors can be computed; given the distant function $d(x, y)$, where x and y are feature vectors composed of N features, $x = \{x_1, x_2, ..., x_n\}$, $y = \{y_1, y_2, ...y_n\}$ the Absolute distance in equal to (eq. 3.3) and the Euclidean distance is equal to (eq. 3.4)

$$d_A(x, y) = \sum_{i=1}^{N} |x_i - y_i| \tag{3.3}$$

$$d_E(x, y) = \sum_{i=1}^{N} \sqrt{x_i^2 - y_i^2} \tag{3.4}$$

Given methods to calculate distance between two feature vectors, in order to determine which class a query feature vector belongs, distance among the query vector and feature vectors in the training set are computed. The class containing the greater number of closer feature vectors to the query feature vector is the class that will be assigned to the query feature vector. Feature extraction is a very important task in pattern recognition. Many features can be extracted from images such as edges, corners and many other ones depending on the aim of the classification. One of the feature extraction techniques used to represent hand gesture data is the computation of the invariant Hu moments.

The computation of the invariant Hu moments extracts seven numbers from an image; these numbers are results of seven nonlinear functions defined on regular moments which are translation scale and rotation invariant. A moment or contour moment is a characteristic of an image contour computed by integrating over all the contour pixels; a contour moment $m_{p,q}$ is equal to eq. 3.5

$$m_{p,q} = \sum_{i=1}^{N} I(x,y)x^p y^q \tag{3.5}$$

where p is the x-order and q is the y-order. Invariant Hu moments are combination of the central moment $\mu_{p,q}$ calculated as (eq. 3.6)

$$\mu_{p,q} = \sum_{i=0}^{N} I(x,y)(x - x_{avg})^p (y - y_{avg})^q \tag{3.6}$$

where $x_{avg} = m_{10}/m_{00}$ and $y_{avg} = m_{01}/m_{00}$. Invariant Hu moments are calculated from normalized moments $\eta_{p,q}$ (eq. 3.7) that allows the moments to be scale invariant.

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{00}^{(p+q)/2+1}} \tag{3.7}$$

In the end the seven linear Hu moments are equal to eq. 3.8

$$
\begin{aligned}
&\phi_1 = \eta_{20} + \eta_{02} \\
&\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
&\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
&\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
&\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2)) + \\
&+(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
&\phi_6 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
&\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
&-(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
\end{aligned}
\tag{3.8}
$$

The invariant Hu moments so calculated represent a feature vector of seven dimension used to describe an hand gesture image.

3.4 Game Design

Having set the design for the gestural interface a study on the game that will meet hand users' inputs is also disclosed. The concept of the game chosen is inspired to an old classic board game called the Labyrinth wood maze (Fig. 3.5). The aim of the game is to tilt the maze wood board to make the ball roll through the maze till it reaches the hole; the board tilt is modified turning the two knobs, commanding respectively the board Pitch and Roll tilt angles, on the Maze sides. Adapting this classic game to a 3D videogame a few considerations on the gameplay and the concept itself should be taken into account. The videogame should result realistic to the users who should interact in the most natural way perceiving the game as realistic as possible. In order to satisfy these requirements the 3D game camera position in the 3D space is chosen

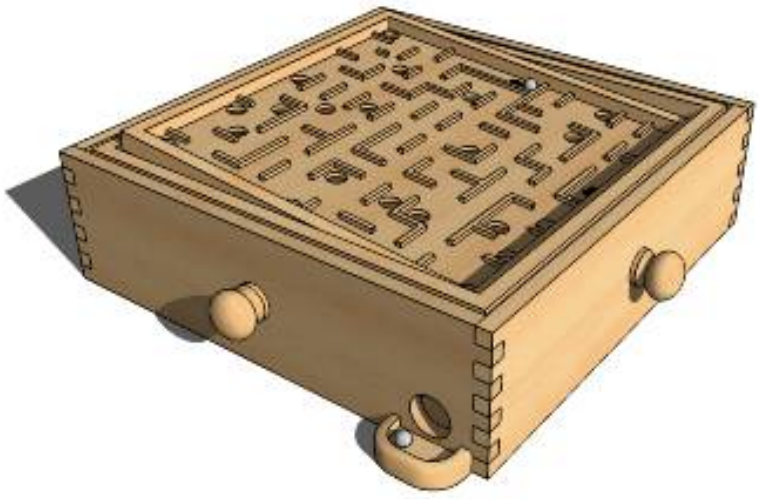


Figure 3.5: Wood Maze. *Copyright Google 3D warehouse, Labyrinth Wood*

just above the Labyrinth game board with its frustum pointing to the maze board allowing a top view of the maze to be rendered on the screen (projector). A top view of the maze board allows the users to perceive the maze lying on the table top surface. The two tilting knobs in the classic game need to be substitute with different hand game inputs, respecting the requirements that hand gesture have to take place outside the projected area, therefore the top maze view. This goal has been reached virtually connecting every user's hand with a different board corner. The hand interaction has been decided along the long side of the projected area, namely along the board long side; in detail every player's hand can slide from the farther corner to almost half of the game area. The board corner close to every hand is virtually connected to it (Fig. 3.6); in effect when the hand slides up towards half of the game long side, the connected corner tends to follow the hand raising the board itself toward the hand and tilting the board towards that hand. On the contrary when the hand slides back towards the corner, it tends to flee the hand lowering the board and tilting it away from the hand. Hands movements can so command the bard roll and pitch angles. Noteworthy point

is that the board always rotates around its centre without translations. The original classic maze game was designed for a single player. The 3D hand gesture maze instead is designed to be played by two players at the same time. The single player hand interaction occupies only half of the game board, thus the second player uses the same hand interaction on the other side of the board using the remaining board part. In the multiplayer mode both players can tilt the board opposing each other inputs having a different ball each.

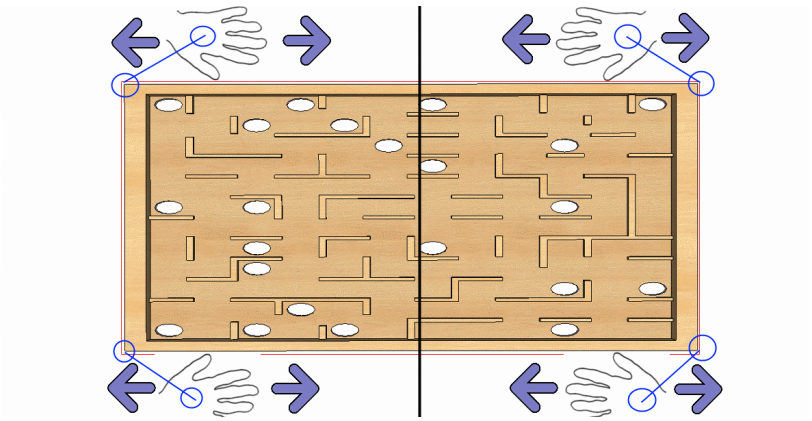


Figure 3.6: Hand interaction

3.4.1 Maze Game Physics

Physical simulation is one the most important part of the maze game, entirely based on physics laws. The ball is rigid body with kinematics properties. The Ball is subject to Gravity force and friction with the board and air; moreover the ball gain acceleration depending on the board orientation and consequently velocity and rotation. The rigid body physical properties are all subjected to physical law studied and described by Isaac Newton. The physicist described the body behaviours in three main physics laws:

- First Law: Every body remains in a state of rest or uniform motion (constant velocity) unless it is acted upon by an external unbalanced force.

- Second Law: A body of mass m subject to a force F undergoes an acceleration a that has the same direction as the force and a magnitude that is directly proportional to the force and inversely proportional to the mass.

- Third Law: The mutual forces of action and reaction between two bodies are equal, opposite and collinear.

The first law means that every object in movement owns a movement vector or velocity value and it does not change unless some other force are exerted; for example the ball would continuously roll if no friction with air and board were exerted on it. The second law means that the force exerted on the rigid body is equal to $F = m * a$ where a is the acceleration which is equal to the change of velocity divided by the change of time $a = dv/dt$. The third law means that whenever a first body exerts a force F on a second body, the second body exerts a force -F on the first body. F and -F are equal in magnitude and opposite in direction. This last law is very important to calculate collision impulses. All the law and their consequences will be implemented in the Maze game to recreate physical realist behaviour.

Chapter 4

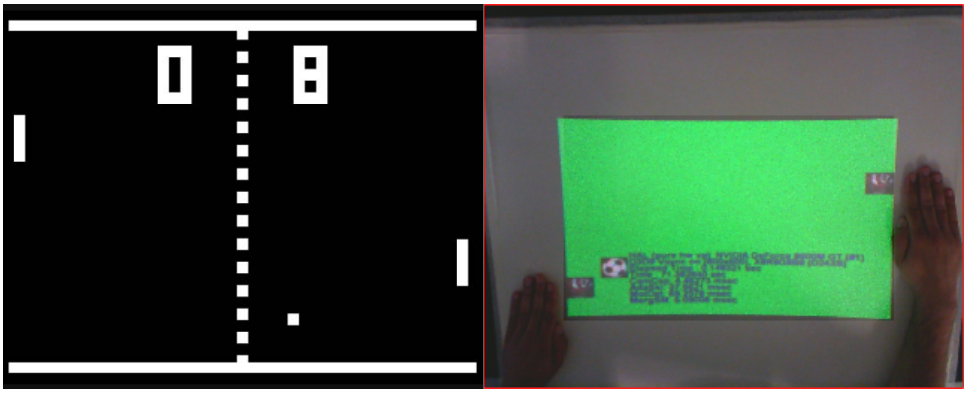
First Prototype

4.1 The Prototype: Pong

Prototyping is a useful development process as it helps to face in advance obstacles or issues that are not highlighted in the design part. In this case a fast prototype was helpful to explore the technology projector-camera discovering in advance its qualities and limitations. In order to achieve it in a short period of time, a simple game playable through hand gestures has been developed; the game chosen is inspired by an old classic game called Pong (Fig 4.1a). The game Pong consists in two paddles (each paddle is driven by a player), one opposite to the other; the aim of the game is to make the ball falling outside the opponent's screen area. In the prototype the game was implemented allowing each player to command his own paddle sliding up and down his hand along the sides of the projected game (Fig. 4.1b).

4.2 Game Stage Configuration

A first important step toward the solution was the configuration of the camera, projector and the choice of a game surface. For the implementation of the Pong prototype



(a) The Classic Pong Game (b) The Pong Prototype

Figure 4.1: The Pong Prototype

camera and Pico-projector have been coupled together; the camera and the projector have been placed one meter height from a white table top surface having both lenses pointing down toward the table top (Fig. 4.2). The table surface has been chosen white to increase the brightness of the projected image, obtaining as result more vivid colours. The camera frustum was set able to capture the entire widescreen projected image and the area around it where the player's hands interact. A screen for blocking out external light has been set above the projector and camera coupled together. Excessive external light in fact makes harder to cope with skin detection and other vision techniques. The Game has been implemented in C++ using DirectX 9.0c as Graphics API; OpenCv 2.0 library has been used for the vision part. The Pico-Projector used was Macrovision ShowwX and the camera was Microsoft Lifecam Cinema camera.

4.3 Challenges and Solutions

The main challenge in making the Pong game was surely the hand segmentation. Difficulties of this task are represented by the nature of the game environment in which the hand interaction is immersed; in fact any attempt to segment hands within

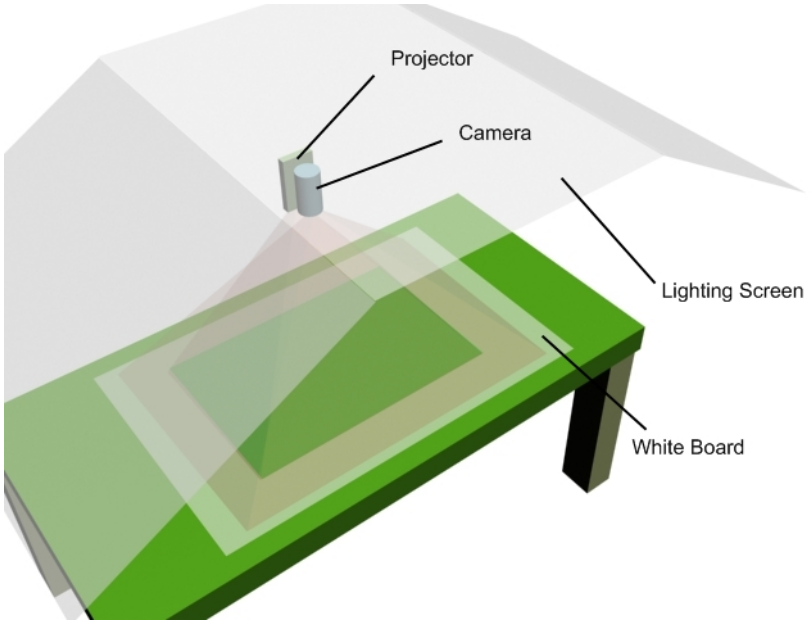


Figure 4.2: Camera-Projector configuration

the projected game area has resulted not sufficient to good hand segmentation. The projected image in fact changes the hand colours and adds noise to the hand shapes making the hand segmentation hard to implement. For these reasons the player's hand interaction has been set outside the projected game area along its sides.

Since the hand interaction takes place outside the projected area, a helpful image processing simplification is to eliminate the game area from the camera captured frames; this trick eliminates a redundant image processing within the game area making the overall computation less heavy. The first subtle issue faced, in isolating captured frame from the game area, was that the camera and the projector do not have respectively the capture Rate and the Render Rate synchronized; the camera in fact adjusts its capture Ratio respect to the external light conditions, instead the projector has a constant Render Rate; the consequence of this not synchronized frame rate is that a frame captured from the camera does not contain a full projector rendered frame; it makes difficult the subtraction of the game area from the camera captured frames.

Averaging over two consecutive captured frames was the solution to isolate the game area. The average image in fact shows the entire game area.

A first attempt to segment hands has been done using the Adaptive Skin Detection proposed by Farhad Dadgostar [13]. This algorithm can produce better skin detection that static skin model algorithms; it analyze the Hue histogram in HSV colour space of input image maximizing the accuracy of the Hue threshold based of motion detection techniques; the motion information are used to refine the Hue thresholding according to the assumption that candidate skin pixels are more likely to be skin ones if they are moving. The algorithm takes as input a colour image and returns a Grey-scale image of skin probability. The probability distribution highlighted precisely skin regions but still was affected by a considerable amount of false positive skin pixels. In order to further maximize the accuracy of the adaptive skin detection, a threshold operation on the skin probability image has been applied. The threshold value has been automatically chosen based on Ground Truth information of an image with skin pixels marked. The image with the skin marked has been compared with the adaptive skin output probability image having in input the first image without skin marks. From these two last images, for each possible threshold value, values of True Positive (Tp), False Positive(Fp), True negative (Tn) and False Negative (Fn) were calculated. The accuracy (A) of threshold values has been calculated by (eq. 4.1).

$$A = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{4.1}$$

Results of the Threshold operation did not satisfy the aim of refining the skin detection though; only after has been noticed that the adaptive skin detection assigns maximum probability of skin to noise regions letting the probability of real skin pixels spread along the entire probability Histogram. The consequence was that not a unique maximum accuracy threshold value was available. An inverse Threshold of the skin probability image was instead useful to eliminate skin noise pixels with high skin probability.

Although the inverse threshold increases the accuracy of the skin probability image, some other technique should be applied to reach a better segmentation.

A motion detection techniques has been taken into account to support the adaptive skin detection in segmenting hands; the main assumption was that only moving blob are taken into account as candidate skin blobs. The method used is the Lukas-Kanade [20] Optical flow Estimation. This algorithm takes as input two Grey-scale images and a block size; the two Grey scale images are consecutive captured frames from the camera and the block size represents the number of pixels for which the motion estimation will be evaluated over the consecutive frames; the algorithm output is the velocity of the moving blocks. The output velocity image results quite noisy; in fact noise in input images produce a slow motion that is capture by the Kukas-Kanade optical flow. A small threshold resolved the noise problem showing only faster moving blocks, in this case hands. One of the major drawbacks of the Lukas-Kanade method is that large homogeneous areas moving show little or no motion respect to their boundaries. The consequence of this drawback was that the moving hand area presented holes and broken parts in his velocity images making difficult good hand segmentation. A solution to this inconvenient was to take a higher threshold of the moving velocity images with the result of showing only boundary moving points. Once moving boundary were identified a Convex Hull algorithm has been used to identify the area within the moving boundaries. A Convex Hull is defined as the minimum convex vector set which enclose a set of points.

Once moving blobs were identified the final hand segmentation has been obtained merging them to the skin probability; skin pixels were taken into account only if within a moving blob (convex Hull). The resulting image was a good segmentation of moving hands. A contour detection on the resulted hand blob image has been run to identify the highest hand tips in each blob. The hand tips (respective left and right) have been scaled to the paddle moving area in the game. The result was commanding the game

paddles through the players' hands. A full Image processing pipeline is described in (Fig 4.3).

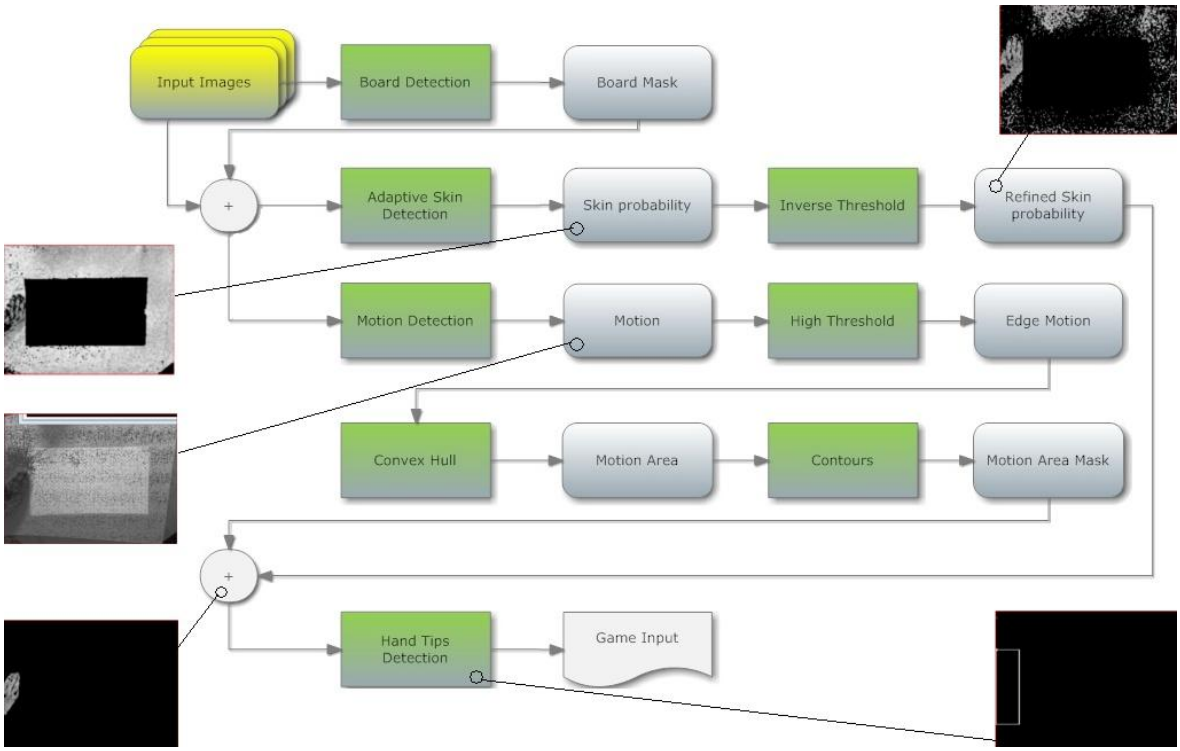


Figure 4.3: The image processing Pipeline

4.4 Discussion

The Rapid prototyping revealed itself as a good solution in the development process; indeed subtle issues such as the capture frame and Render rate not synchronized or the skin probability with high value set on noise have been discovered; a further research will be concentrate in solving this issue. Furthermore experiment results show that a more accurate system has to be developed and a more robust to noise system have to be researched to improve the inputs quality and the game experience. The application Average Frame rate suggests that a multi-thread design should be taken in consid-

eration to separate into two different threads the vision computation and the game one. The vision pipeline appears costly in computation hence a simpler approach has to be studied; indeed every further step reduces the Robustness of the entire system. Moreover a limitation of the hand segmentation consists in the detection of only static hand gestures. Only moving hand gestures are taken into account in Pong prototype; static gesture recognition can increase the game experience augmenting the set of input gestures available for the game. Other subtle issues involved in hand detection were the shadows produced by the hands on the table; hand shadows in fact produce movement in the scene detected by the motion algorithm; the final hand detection wrongly consider shadows part as hand blob parts. Directions for the final product have been set and preliminary issues have been discovered; the next chapter will report solution to issues raised by the first prototype.

Chapter 5

Final Demo Implementation

Aware of issues raised by the rapid prototype, the final demo implementation addresses them using the fastest best technique to solve the problems. Goals of the final implementation are listed as follow:

- Separate into two different threads the Game Pipeline from The Vision Pipeline.
- Improve the reliability and accuracy of the Game Board Detection.
- Increase accuracy of the Hand Detection simplifying its pipeline.
- Introduce Gesture Recognition using pattern recognition.
- Implement the Maze game for single and Multi-player.

Most of the software structures built for the prototype have been reused; although the vision pipeline and the game object have been re-implemented with different techniques. The implementation disclosure is divided in two main sections; the former describes in detail the new vision pipeline adopted explaining the techniques and relative issues. The latter describes the game implementation with an overview of the techniques and methods used to address the game issues.

5.1 Game Stage Configuration

The game stage configuration follows the design description and the prototype one. Few improvements have been introduced in the technology settings, removing the few environment constraints introduced in the prototype part. The lightning screen has been removed improving the vision system algorithms to cope with ambient lights. The white board lying on the table surface used in the prototype has been removed as well allowing the system to be more flexible respect to different table surfaces and materials. The camera-projector has been coupled together at a height of one meter on photographer tripod (Fig. 5.1).

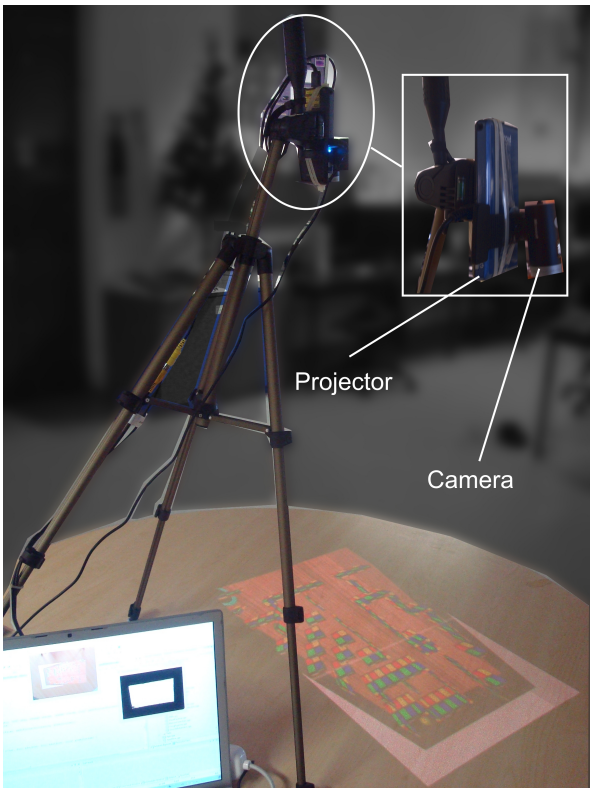


Figure 5.1: Final implementation Game Setting

5.2 Vision Pipeline

The Vision Pipeline for the final implementation had been strongly simplified (Fig. 5.2). The prototype pipeline (Fig. 4.3) presented many steps; every step like in a chain introduces a weak point; in fact each step was aimed to address a particular issue but it also brought up technique related drawbacks. The prototype pipeline appeared quite slow and slightly weak in Robustness and Accuracy. In particular the block of Skin detection and motion Detection have been replaced by an Adaptive Background Subtraction technique in the final implementation. In fact the skin detection resulted highly light dependent; the skin color in fact changes respect to the light that reflects on it; even implementing adaptive skin detection, skin blobs were wrongly segmented missing actual skin parts detecting instead background parts. In order to solve this problem a motion detection was implemented in support of the only skin detection. Reliable skin blob were taken into account only if moving at a certain speed. Direct consequence of this assumption was that only moving gestures could be recognized adding unnecessary constraint to the gameplay. The new implemented pipeline (Fig. 5.2) is composed of only three main modules: Board Detection, Adaptive Background Subtraction and Hand Detection. The whole pipeline is executed on a different thread from the game pipeline. The vision pipeline thread share only hand detection position data with the game separate thread. The two main pipelines have been separate to improve performance that during the prototype revealed to be quite slow for a reactive real time application. The vision pipeline is execute every twenty milliseconds while the game thread is executed as much as possible; reason for executing the vision pipeline in time interval is that the camera capture rate is slower than the game thread execution rate. The game board detection step is executed at system initialization and once the game board area is detected the step is skipped; the remaining two step are executed every frame according with the vision pipeline execution time interval. Each step shown includes internal pipelines which are disclosed in the following paragraphs.

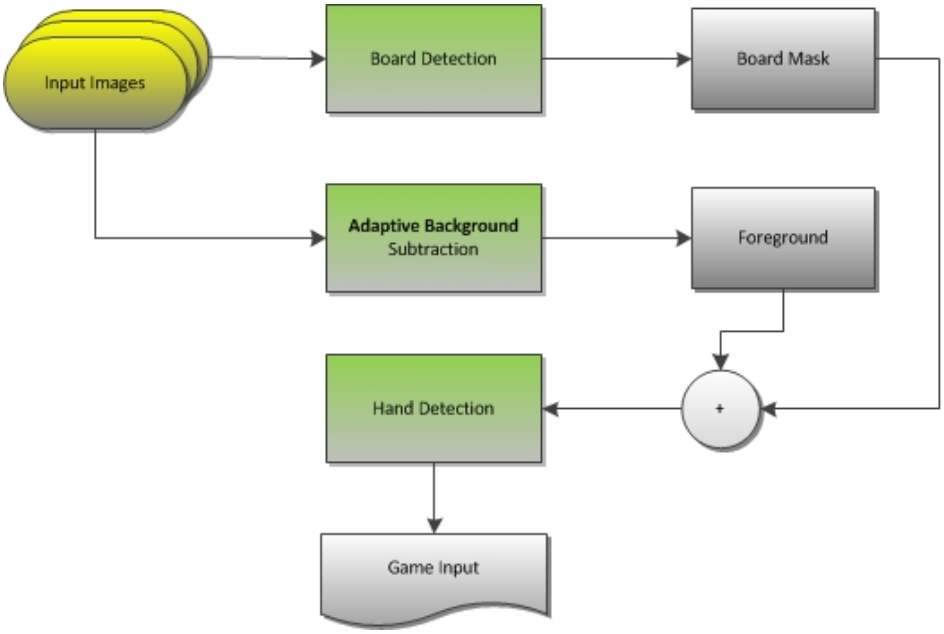


Figure 5.2: Final implementation Vision Pipeline

5.2.1 Game Board Detection

The Game board Detection component in the Vision Pipeline is responsible of recognizing and locating the projected image area on the table within the input camera frames. The Game Board Detection is extremely important for the correct operation of the all other modules and the game experience. Once the game area is recognized, a game area mask can be created which will be subtracted to the input images before further processing; doing so, the pixels within the game area will be ignored in further processing saving computational power and time. Besides, given the game area perimeter it is possible to identify areas along the long side game area where hands are suppose to interact; Moreover only knowing the game area in the input frames is possible to scale the input from "image space" to "game space". The Game Board Detection was implemented in the prototype by Canny Edge Detection. This technique has revealed slightly weak to address the problem; in fact for the 50% of the detections the game board was missed or wrongly detected; in addiction it was quite sensitive to

lighting conditions, producing unstable results. The Board detection has been completely re-implemented implying several different techniques shown in the Fig. 5.3. One important prerequisite of the board detection was a process called Camera Cali-

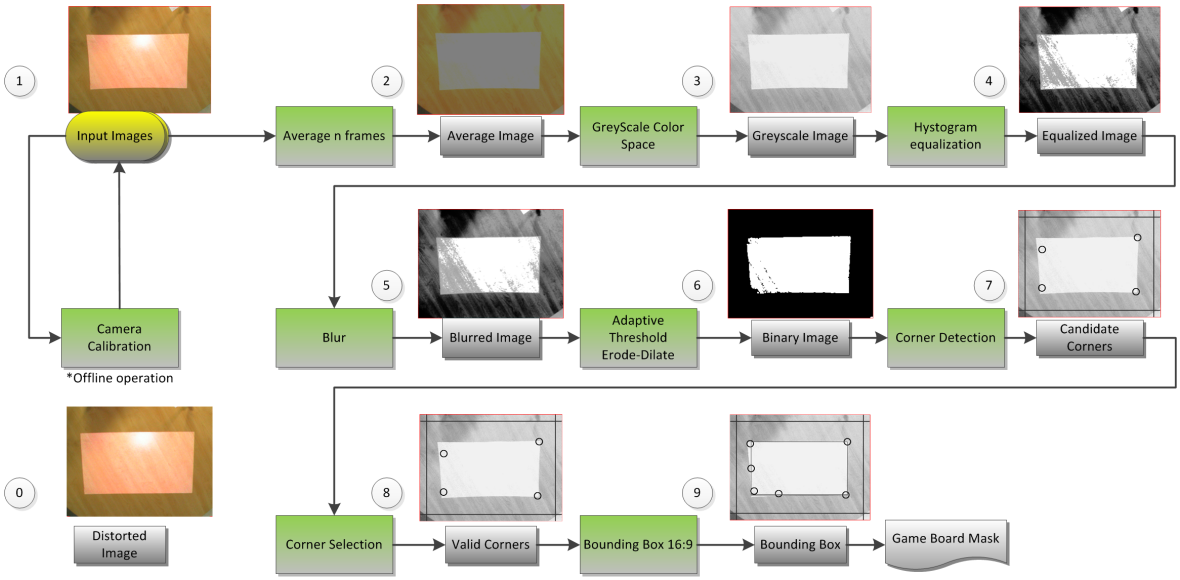


Figure 5.3: Game Board Detection Pipeline

bration; in fact camera lenses have defects such as Curvature of field; this defect leads to geometrical distortions in the camera captured frames. Curvature of field expresses the phenomenon that the image of a plane perpendicular object to the optical axis on the object side of the lens projects to object a parabolical surface. If the object on the object side is a flat grid consisting of squares then it is projected either as a barrel or pincushion [22]. In Fig. 5.3 step 0 it possible to see that the camera used for the project was affected by Curvature of field. The camera calibration is the process that finds the camera intrinsic parameters and a distortion vector used to remove geometrical distortion from input images. The camera calibration process has been execute once saving the camera intrinsic parameters in XML files loaded by the system and applied to the input images.

In order to detect the game board area, a white image is projected onto the table.

Reason why a white image was projected was that the white colour has the highest colour intensity and allows the camera to capture a better contrast between the board area and background. No hands or other extraneous objects interaction are allowed during the game board detection phase, otherwise a wrong detection would take place. The frame averaging step (Fig. 5.3 - step 2) consists in capturing n frame and averaging their pixels value over the n frames in an output image. This step was necessary to compose a frame containing the entire game board image since the camera frame rate and the projector one are not synchronized; besides the averaging step helps to minimize the light reflection that affects the table top. A colour space conversion from RGB to gray-scale of the averaged image has been executed to simplify the image and save computational power processing having only one channel. The image at this stage shows clearly the game board area; it is easy for a human eye to recognize it but not enough for image processing algorithms. A Histogram equalization operation has been applied to the gray scale image in order to improve its contrast (Fig. 5.3 - step 4). The Histogram equalization is aimed at stretching out the histogram distribution of the entire intensity image range; often in images the histogram intensity values are clustered in a certain zone of the available range; this clustered distribution is cause of images with poor contrast. The resulted image after the histogram equalization shows a higher contrast, although some zones within the game board area appears to be degraded with background pixel intensities. A smoothing operation, using Gaussian Blur, has been applied to the equalized image to make more homogeneous the intensity within the game area (Fig. 5.3 - step 5). The Blur operation in fact computes each output pixel as the mean of all the pixels in a window around the corresponding pixel in the input image. At this stage, based on the assumption that the game area occupies most of the pixels within the Game board area, an "adaptive threshold" operation on the blurred image is applied (Fig. 5.3 - step 6); the threshold is defined adaptive because its value is based on the mean intensity value of the blurred image plus an empirical

fixed offset. Morphological operations are applied to the threshold image to remove noise and narrow features such as bridges between larger groups of points. After the threshold operation a binary image is obtained presenting with white intensity pixels the game board area and with black intensity pixels all the rest. A corner detection algorithm is run on the threshold image (5.3 - step 7) in order to find at least three corners and at most four corners forming an AABB bounding box. The corner detection is aimed to find the main corners of the board area; the definition of a corner is defined as an intersection of two edges, but in computer vision usually is referred to an interest point which can be located robustly which can include corners but also small marks. Many corner detection techniques are available; for the board detection pipeline the corner detection chosen was the Harris corner detector. Harris detector is based on the local auto-correlation function of a signal; where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions. It has been chosen for its strong invariance to rotation, scale, illumination variation and image noise [10]. Due to the nature of the threshold image the Harris corner detector finds many candidate corners; in fact the threshold image presents many irregularity that are detected as corners. The corner selection steps and are responsible (5.3 - step 8,9) of filtering all the potential corners found by the corner detection in a valid subset. In particular corners which lie within ten pixels from the image margins are ignored. The potential other corners are sent to the bounding box component which takes a four corner combination and calculates an AABB bounding box on it. An Axis-aligned minimum bounding box (AABB) for a given point set is its minimum bounding box subject to the constraint that the edges of the box are parallel to the (Cartesian) coordinate axes. Once a bounding box is calculated, it has to pass a test; in fact any combination of four corners can generate a bounding box but in this system the game area bounding box must have side ratio equal to 16:9 (letterbox or widescreen) such as the projected image. If the bounding box satisfy this condition

the Game Board area is considered detected and a binary mask image is generate and stored during all the system activity. The mask presents pixels with intensity values set to zero within the game board area and to one outside it. Such a mask is useful because multiplied to other images from the camera set to zero all the pixels within the game area isolating it from further processing. Information from the bounding box are also stored because useful in following vision pipeline steps. The overall Game Board Detection is executed at the system initialization and it re-execute itself until a valid bounding box is found and is no more executed after; the system does not execute any other pipeline block until the game board detection is correctly detected. The re-implemented pipeline for the game board detection recognizes board with an accuracy of 90% hence it results 40% more accurate of the prototype one. Although the game board recognition results sensitive to light changes; in fact it works out better with an reasonable amount of light like in a indoor environment with suffuse light and not direct light to the table top surface. If the amount of light increases or decreases the accuracy will strongly decrease; a solution to this inconvenient is a direct intervention on the camera exposure. Specifically when the light conditions result darker than the described optimal conditions a camera exposure decrement would bring up the detection accuracy back. In the opposite case an increment on the camera exposure would fix the issue.Indeed the camera exposure limits the amount of light allowed to fall on the photographic image sensor.

5.2.2 Background Subtraction

The Background subtraction module is responsible of separating foreground and background from input images; in this system the background is represented by the table surface and the game board area, while the foreground is represented by all the rest, namely hands sliding along the long sides of the game board area. The Background subtraction module (Fig. 5.4) is composed of three main sub-modules: the background

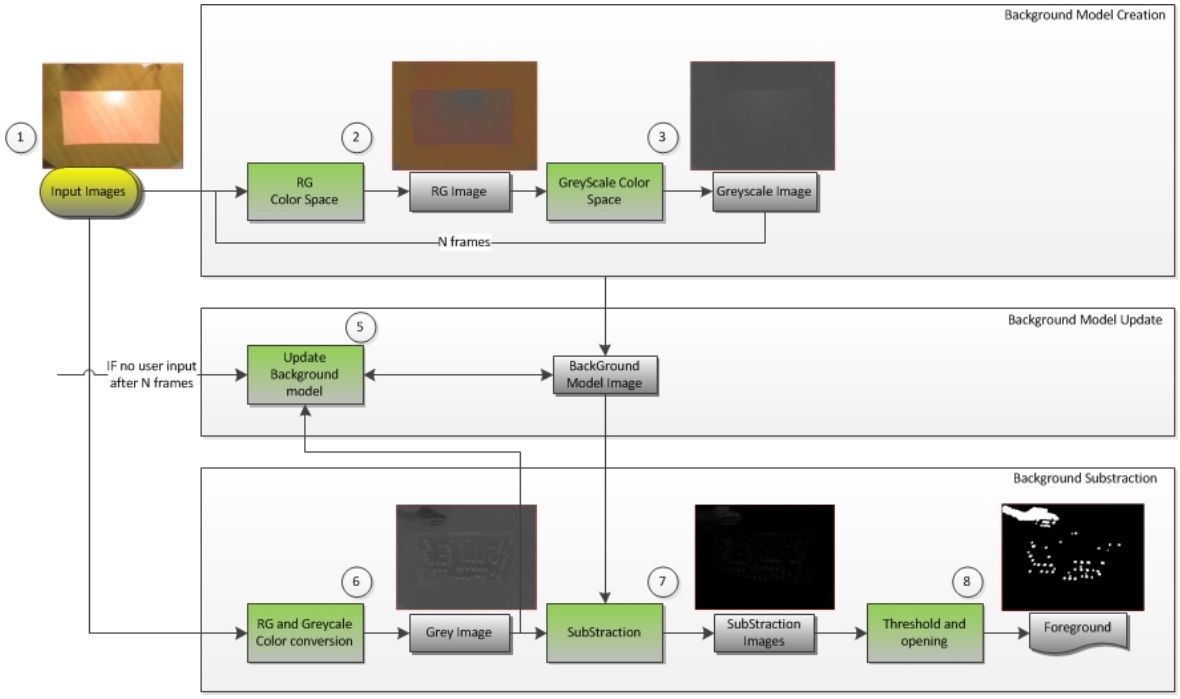


Figure 5.4: Background subtraction Pipeline

model creation, the actual background subtraction and the update background model. The background Model creation (Fig. 5.4 - steps 2,3) is responsible of creating a background model image; the background model is an image that describes as close as possible the background; in fact the background model is not composed of only one frame of the background but by a composition of more than one input frame. A strict assumption is represented by the impossibility of user interaction during the background model creation; a hand interaction during the model creation can degrade the background model including in the model not stationary elements. Creating a reliable background model is of fundamental importance because each input frame (adequately pre-processed) is subtracted by the background model image in order to obtain the foreground image, thus a wrongly created model would lead to wrong foreground segmentations. The background model creation accumulates N frames; each input frame accumulated is converted by RGB colour space into rg Chromaticity colour space (Fig.

5.4 - steps 2). The rg Chromaticity is a colour space that has only two colour channels calculated as combination of the RGB channels (eq. 5.1 and 5.2).

$$r = \frac{R}{R+G+B} \tag{5.1}$$

$$g = \frac{G}{R+G+B} \tag{5.2}$$

A reason for using this colour space is that through the colour conversion the image luminance component is removed. Consequence is that to every pixel is assigned a colour value regardless how colourful the original colour is. An issue raised during the prototype part was that hand gesture shadows interfered with the hand segmentation and the motion detection reducing the accuracy of the final result. Using rg Chromaticity colour space the shadows interference is minimized.

The rg Chromaticity frames are subsequently converted in greyscale colour space to make computation faster (Fig. 5.4 - steps 3). Once N greyscale frames are accumulated the background model image is created setting every pixel to the median value of the N greyscale frame sorted pixel values. This operation consents to capture a better background model not based only on one frame; in general each consecutive frame presents slightly different colour information even if the camera is steady capturing a static scene; these colour changes are due to many factors but mainly noise and number approximations in the digital images.

Once a background model image is generated, the creation module is not executed anymore during the entire system activity; instead for every input frame the actual background subtraction is executed. The input RGB frames are converted before in rg Chromaticity colour space and then in grayscale colour space (5.4 - steps 6) to meet the same background model format. The operation of absolute difference between the grayscale input frames and the background model is executed for each frame. The

absolute difference operation, given two greyscale images, calculates the absolute difference between each pixel of the first image with each pixel of second input image; result is a third image also greyscale representing the foreground. The Fig. (5.4 - steps 7) correctly shows the foreground of the image (a hand); parts of the game board area are also recognize as foreground because when the model was created the projected image in the game area was only a white image; during the game the projected image changes constantly rendering game objects which results different from the background model, namely recognized as foreground. This issue does not represent a big problem because being aware of the position and the dimension of the game area bounding box, those pixels are in the foreground image can easily be ignored in the game hand detection module. Furthermore the absolute difference will present some noise in its result; the noise is represented by minimal difference between pixels representing background (pixels representing foreground have a greater difference). The background noise it is eliminated by a threshold operation (5.4 - steps 8) followed by a opening morphological operation.

One of the biggest limitations of the background subtraction is that the background model is not dynamic. During the activity of a system changes in lighting conditions are a threat to the accuracy of the subtraction because the model will not reflect any longer the real background colour. Rg Chromaticy colour space conversion minimizes this problem but for big lighting changes the subtraction accuracy will be affected. Other threats to the background subtraction accuracy are extraneous moving object in the scene that becomes stationary in the scene itself. These objects with a static background model would be always recognized as a foreground. Solution to these threats is a dynamic background model; this kind of model can be update during its activity assuring a correct background subtraction. The background subtraction pipeline provides a background update sub-module (5.4 - steps 5). The background update sub-module takes as input a camera frame, converted before in rg Chromaticity

and the in greyscale, and the background model image; if the first image is labelled as a and the second as b and with $a(i, j)$ the value of the pixel at (i, j) position within the image, the background model execute the following operations (eq. 5.3).

$$b(i,j) = \begin{cases} b(i,j)+1 & \text{IF a(i,j)} > \text{b(i,i)} \\ b(i,j)\text{-}1 & \text{IF a(i,j)} < \text{b(i,i)} \end{cases} \tag{5.3}$$

The pixels of the background model are incremented or decremented by one respectively if the input frame has greater or smaller intensity value at the same pixel. This computation resolves the threat of moving object becoming stationary in the scene. In fact if an object enter the scene the value $a(i, j)$ will be greater of $b(i, j)$ so the value $b(i, j)$ be will be incremented by one; if the object stays in the same position frame by frame it becomes part of the background. On the contrary if a stationary object leaves the scene or simply moves it will be gradually erased from the background.

Implementing an update method resolve the accuracy threat to static background model but result contradictory when stationary object needs to be recognized in their stationary position. For example, during a user interaction with the game, user's hands can be steady to maintain a certain input into the game; leaving the update algorithm running for every frame, the user's hand would become a background object creating false input whenever the hand moves again. Solution to this problem was to execute the background model update only if no user inputs were registered for N frames. In this way the background could be update only when no user hand interaction is happening. The behaviour of this update is better explained in following section where the hand detection mechanism is explained in detail as well.

5.2.3 Hand Detection

The hand detection module is responsible to detect the position of the four possible hands (two for each player). The Hand detection module is strongly coupled with

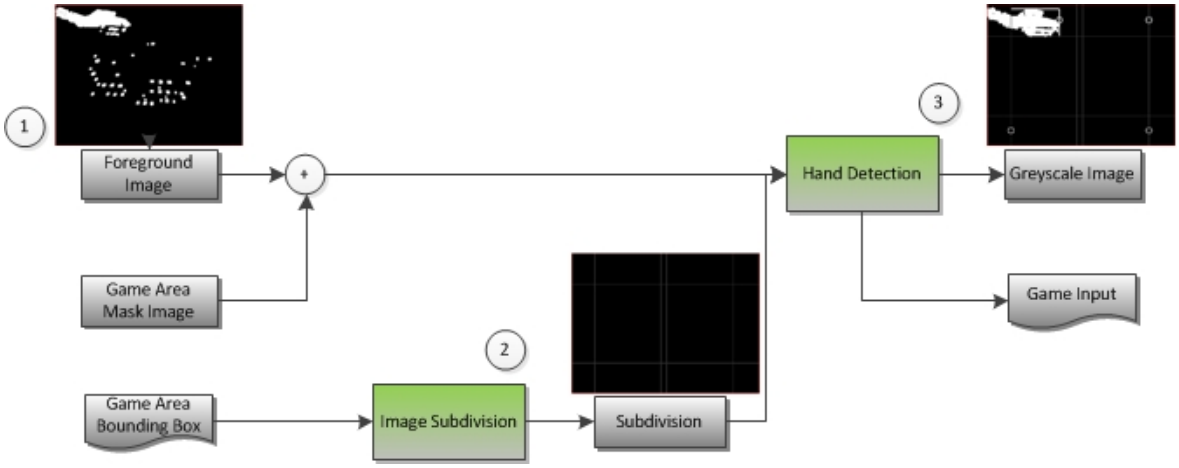
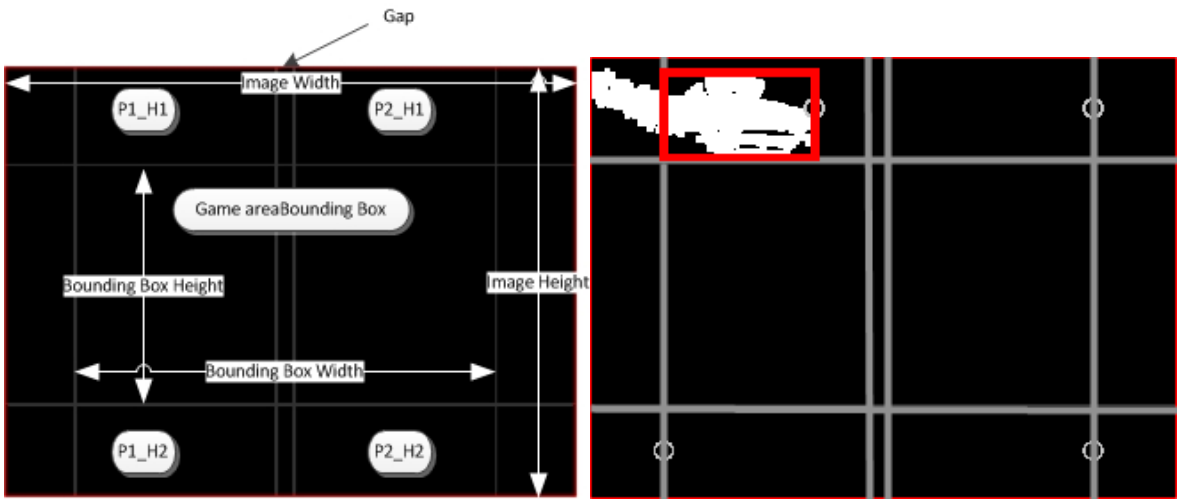


Figure 5.5: Hand Detection Pipeline

the other modules; in fact, the hand detection pipeline (Fig. 5.5) needs as inputs, the foreground image, the game area mask image and the data from the game area bounding box. The first step in the Hand detection pipeline is the Image subdivision (Fig. 5.5 - step 2). The image subdivision is aimed at defining areas within the foreground image where the hand interaction will take place. The Fig. 5.6a describes the subdivision areas. The areas $P1\_H1$, $P1\_H2$, $P2\_H1$, $P2\_H2$ (the acronym Px_Hx



(a) Foreground frame subdivision (b) Hand Bounding Box

Figure 5.6: Hand detection

stands for Hand number x of the player number x) represent the four areas where hand interactions are taken into account. The central big rectangle represents the game area bounding box. The hand areas are calculated from the width of the game area bounding box. All the other areas are ignored from hand detection. The narrow gap area is aimed to divide the hand areas of the first player by the hand areas of the second one. Once the image subdivision is set the hand detection module starts to check for hand presence in each area. The hand detection step (Fig. 5.5 - step 3) takes as input the foreground image and the game area mask image; these two images are multiplied together in order to set to zero all pixels within the game board area in the foreground images. After this step for each hand area every white pixel is counted; if the number of pixels within an hand area is greater than five hundred pixels a bounding box of all the counted pixels is calculated; the hand bounding box calculated represent the detected hand (Fig. 5.6b). The filter on the number of pixels within a certain hand area helps to cope with noise pixels that occasionally appear in the foreground image. Once information from hand bounding box are available the position of hands tips are extracted (Fig. 5.6b circles in the picture); if no input changes are detected the hand tip values are set to their previous values. Due to the nature of the computer vision detection the hand tips values at one particular frame is result of the mean value over the respective hand tips three previous values; the mean values stabilize the hand tips positions minimize their jittering; in fact the hand tip detection of even a steady hand would have a minimal displacement in its position frame by frame. The hand tip position is calculated and stored in only one dimension, that is the horizontal one (Cartesian x axis); only one dimension is sufficient to gather input information; before the detected hand tip positions are sent to the game module they are transformed into Hand area local space; in effect the hand tip position is detected in Image space which range $[0, ImageWidth]$. The hand detection is strictly coupled with background subtraction update module; in effect the background model update takes place only if

there is no change in any of the hand tip positions after N frames.

5.2.4 Hand recognition

The hand detection module detects hands moving along the long sides of the game board area quite fast and precisely. It works out good with the assumption that only hand can move along the game area sides; if instead other object from the foreground move in the scene they will be recognized, still precisely, as hand inputs. An attempt to recognize different kind of hand gestures within hand blobs has been made exploiting a pattern matching technique. This kind of technique helps to recognize hand gestures from a training set of data; the training set is composed of vectors; all the vectors in the training set must have the same dimension; each vector in the training set is assigned to a labelled class. The algorithm can accept any N vectors with any X dimension; a good modus operandi tells that a reasonable amount of vectors for each class should be given as training set to allow the algorithm to better separate classes. Aim of using a

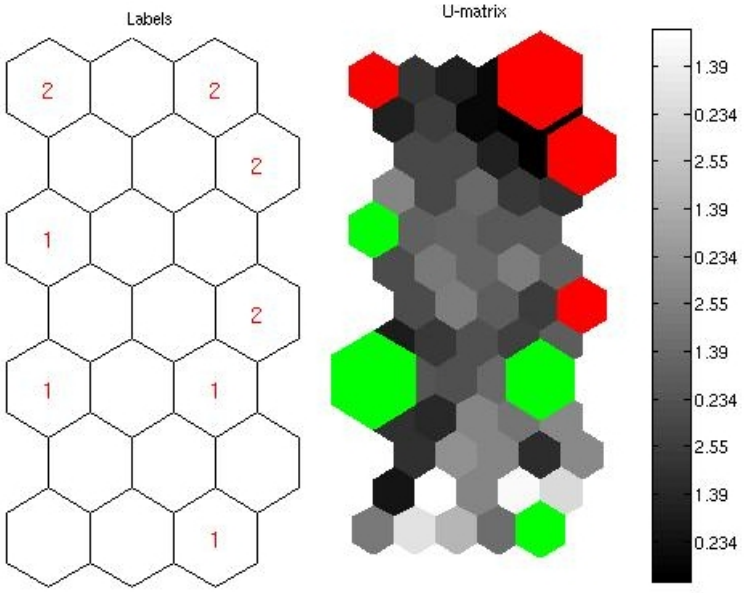
(a) Fist Gestures

(b) Hand Back Gestures

Figure 5.7: Hand gestures training set

patter matching algorithm in this systems is to identify whether a hand bounding box (Fig. 5.6b) represent a full hand back or a close fist. The pattern matching algorithm chosen was the K-Nearest Neighbors (KNN) because it represents one of the simplest and fastest classification algorithm. A challenge for patter recognition techniques is the choice of the vector describing the object to classify. In gesture recognition techniques

these vectors are called feature vectors because they express an image with several numerical values. The vector space chosen was created, given a gesture image, by extracting the Invariant Hu moments [15]; the calculation of invariant Hu moments provides seven numbers for each image provided in input; these numbers are similar for similar images even with dissimilar scale. In order to create a training set seven picture of hand backs (Fig. 5.7b) and seven picture of fists have been taken (Fig. 5.7a). Each RGB picture has been converted in greyscale colour space and equalized in its histogram to improve its contrast as suggested in Yen [12]. A separate software tool has been implemented to create the training set from the fourteen pictures; the tool in fact calculates invariant Hu moment for each picture and assign for each vector extracted a class label 1 for back hands and 2 for fists. The training set is saved into external an XML file describing the training set matrices. The KNN algorithms has been integrated in the Vision Pipeline; on the system initialization the KNN load the XML file describing the training set. Frame by frame whenever an hand bounding box is detected, the same hand bounding area taken from the current input image is converted in greyscale and equalized in its histogram; the region of interest so calculated undergoes the invariant Hu moments calculation obtaining the correspondent feature vector; the feature vector used as query for the KNN algorithm which returns the class membership. Unfortunately during the experiment the two kind of gesture were used while playing the game but the KNN kept on returning wrong membership classes for query hand gestures; occasionally the KNN algorithm recognized the right membership class but the overall result was not accurate enough to be considered stable and to be integrated into the vision pipeline as hand recognition module. A further investigation on the reasons why of this result, were made on the training set itself; being each vector made by seven dimensions a graphic visualization of its hyperspace was not possible with commons tools. A way to visualize the distribution of the two classes in the experiment was a Self-Organization Map; a SOM is a type of artificial neural network



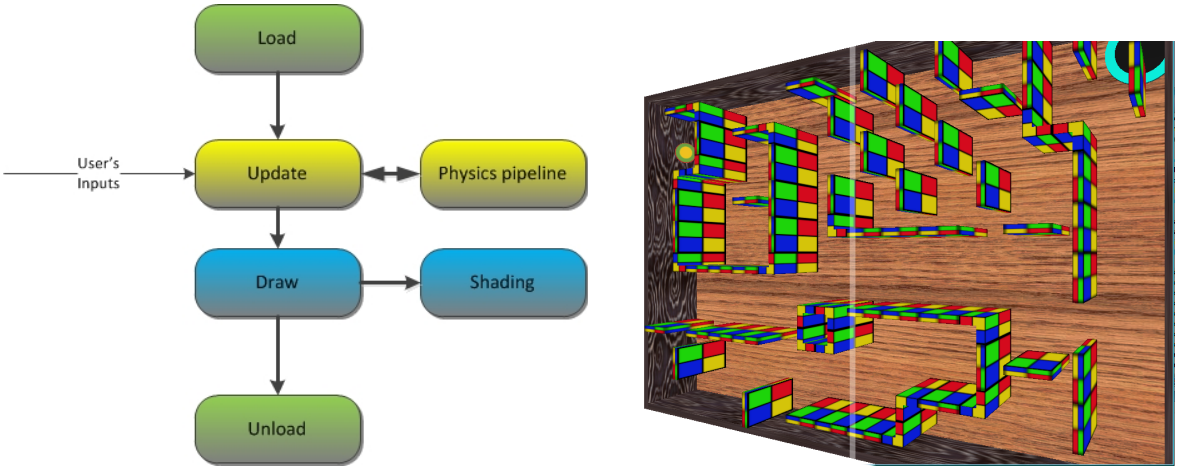
(a) Labeled Classes (b) coloured Classes

Figure 5.8: Training Set Self-organizing map

that is trained using unsupervised learning to produce a two-dimensional discretized representation of the input space of the training samples. The visualization of the hand gesture training set (Fig. 5.8a and 5.8b) shows that the class 1 (hand back gesture, green class) results quite spread out around the hyperspace; the class 2 (fist gesture, red class) results slightly less spread out. The class distributions explain the instability of the recognition; in effect a query gesture would be likely classified in either one class or the other because the separation margin between classes is not wide enough. For these reasons the hand recognition model has not been integrated in the final pipeline.

5.3 Game Implementation

The final Maze game shows a greater complexity than the prototype Pong Game. The Maze game indeed is a three Dimensional game that introduces more complex techniques in the game development such as Real Time Rendering, Real Time Animation,



(a) Maze Game Pipeline (b) Maze game screenshot

Figure 5.9: Maze Game Implementation

Real Time Physics, 3D models creation and etc. The 3D scene is set such that a user playing the game watches the maze from its 3D top view (Fig. 5.9a); the choice of setting the 3D scene to render the maze top view improves the augmented reality experience letting the player believe that the maze board lies on the table surface. The Fig. 5.9a shows the Game Pipeline step subdivision used to render the game on the screen (projector). The load step is responsible of loading in memory external assets such as 3D models, textures, shader effects and to set up the whole 3D scene. The update step is executed every frame and executes all the operation to update the game state for each game object. The draw step render on screen the game objects with the updated parameters using shaders on the graphic card to speed up the computation. The unload step is execute at the application exit and is responsible of clearing all the memory instantiated.

5.3.1 Game entities

The game implementation follows the Object Oriented Programming (OOP) principles using widely polymorphism as shown in Fig. 5.10. Every entity in the game extends

the functionality of the *GameObject* entity; the game *GameObject* in fact defines the basic functionalities that every object in the game must have such as load, update, draw and unload. The *Sprite* entity extend all the father functionalities and defines data such as the texture represented, and its two dimensional world Transform Matrix used to correctly render the image on the screen. In computer graphics a sprite refers to a small two dimensional image that is integrated into a larger scene [21]. *Sprites* in the Maze Game are the *Hand Indicators* and all the rest of *Head Up Displays* (*HUD*). Hand indicators are useful to visually track where the user input is; hands indicator are 2D images which position follows the users hand tips; they improve the user awareness about the hand input effects in the game. A *scene* is a *GameObject* that keeps a list of multiple *GameObjects* or *Scenes* themselves and update and draw them altogether. The *Maze* is the principal game scene which contains all the others game objects and scenes. The *MazeWall* scene contains instead all the maze walls.

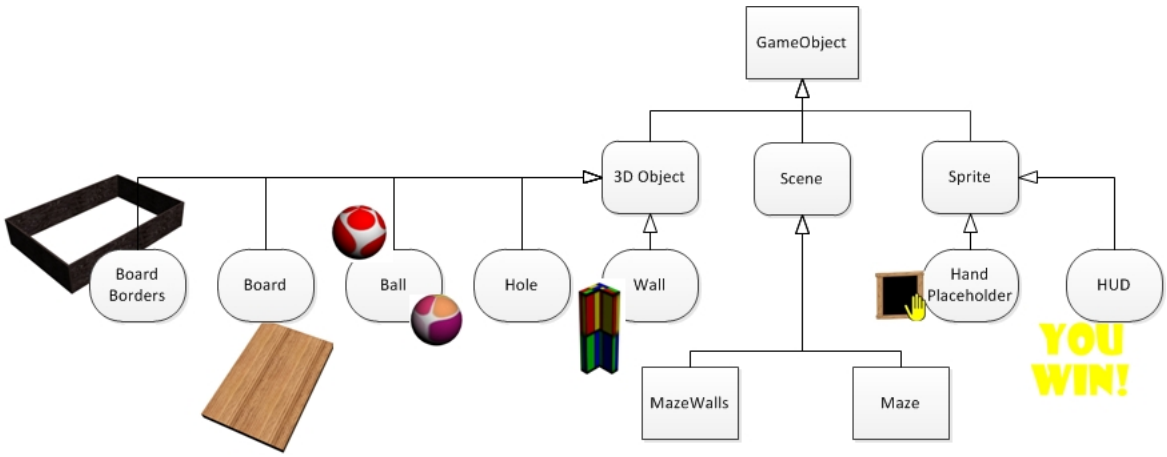


Figure 5.10: Game entities

The *3DObject* entity store information about the three dimensional models such as 3D world Transform Matrix, original geometry composed of triangles and vertexes, and all the lighting values used to correctly rendered the model on the screen. *3DObject* entities are the *board*, the *ball*, the *wall*, and the *board borders*. The *Ball* is the object

that players need to move by tilting the *Board* in order to win the game. *Walls*, *Board Borders* are physical obstacle for the ball movements; in particular the board contrasts the ball gravity force, the board borders avoid the ball to fall off the board; in the end the walls impede the ball to reach the *hole*, goal of the game. The walls can be of two kinds: straight or bend; together bend and straight walls compose the maze that users have to overcome to make the ball falling in the hole.

5.3.2 Game user input mapping

One of the most important implementation parts is surely the user input mapping; the vision pipeline provides to the game pipeline data of hand tip positions for each of the player's hand. This values need to be mapped into the game in two dimension to allow the hand indicators to follow the users' hands in the projected game image. Moreover, assuming that each player's hand is connected to one of the four board corners, every hand input has to be transformed in two rotation angles in a three dimensional space. The angles are used to create rotation matrices to rotate the board and all the other game objects according with the user inputs (Fig. 5.11). Assuming that *HandPos* is the

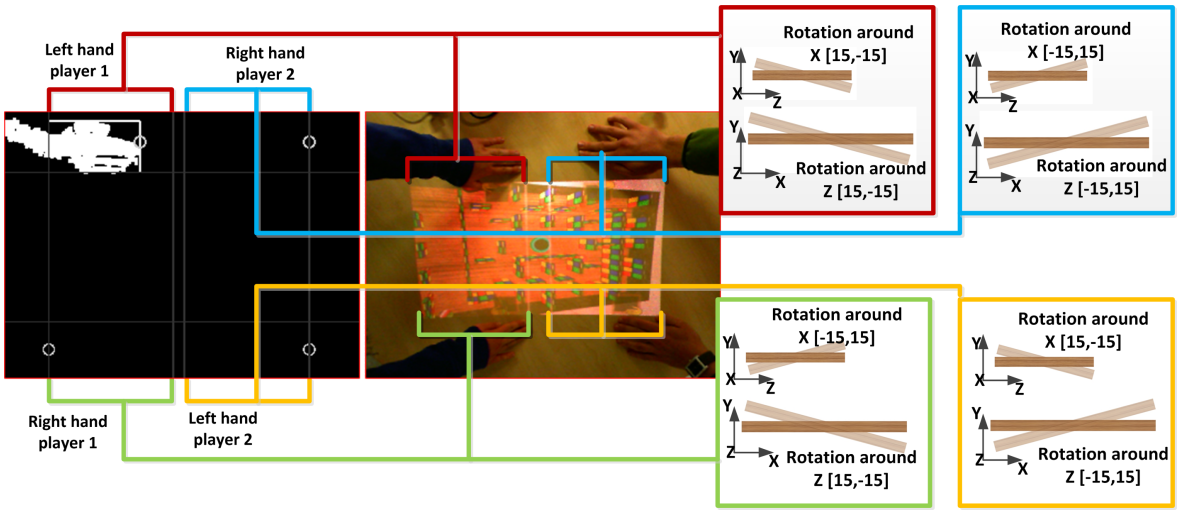


Figure 5.11: User input mapping in the game

position of a hand in hand region local space, *ImageWidth* the full horizontal camera frame width, *Gap* is the gap within the background image subdivision, *ScreenWidth* is the game windows width and *GapScreen* is the *Gap* transformed in game windows proportions, the hand indicator position in screen space (*HandIndPosition*) is equal to (eq. 5.4)

$$HandIndPosition = \frac{HandPos * 0.5 * (ScreenWidth - GapScreen)}{0.5 * (ImageWidth - Gap)} \tag{5.4}$$

Once the hand indicator positions are calculated all the hand indicators sprites can be render correctly on the screen. From the *HandIndPosition* is possible to calculate the value of the rotation angles used for rotating the board in 3D. Assuming that *anglerange* is the maximum angle range, the rotation angle (*RotationAngle*) is equal to (eq. 5.5)

$$RotationAngle = \frac{HandIndPosition * 2 * anglerange}{0.5 * (ScreenWidth - GapScreen)} - (0.5 * anglerange) \tag{5.5}$$

Once the *RotationAngle* is calculated two rotation matrices can be calculated for each hand input; the two rotation matrix calculated are respectively a rotation around the X axis (R_x eq. 5.6) and a rotation around the Z axis (R_z eq. 5.7) of the same *RotationAngle* per hand input.

$$R_x(RotationAngle) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(RotationAngle) & -\sin(RotationAngle) \\ 0 & -\sin(RotationAngle) & \cos(RotationAngle) \end{bmatrix} \tag{5.6}$$

$$R_z(RotationAngle) = \begin{bmatrix} \cos(RotationAngle) & \sin(RotationAngle) & 0 \\ -\sin(RotationAngle) & \cos(RotationAngle) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

Assuming that for each hand input two rotation matrix are calculated; the combination of all the rotation (*TotalRot*) is equal to eq. 5.8, where $R_x H_i P_j$ is the rotation around the x axis and $R_z H_i P_j$ is the rotation around the z axis of the i hand of the j player. The combination of all the rotation *TotalRotation* is passed and applied to all the game entities involved with the board rotation.

$$\begin{aligned} TotalRotation = \quad & R_x H_1 P_1 * R_z H_1 P_1 * \\ & * R_x H_2 P_1 * R_z H_2 P_1 * \\ & * R_x H_1 P_2 * R_z H_1 P_2 * \\ & * R_x H_2 P_2 * R_z H_2 P_2 \end{aligned} \tag{5.8}$$

5.3.3 Maze loader

In order to improve software maintenance and make the creation of different maze combinations easier and faster, the disposition of the walls maze in the game is loaded from an external text file. In order to make this task automatic the maze board has been subdivided in cells, fifth teen rows and ten columns. Each cell can contain only one game object (Fig. 5.12). Game objects allowed within cells are: straight wall, corner wall, hole, first player starting ball position and second player starting ball position. The straight wall can be positioned in two different local orientations: horizontal and vertical. The corner wall instead can be positioned in four different orientations. The

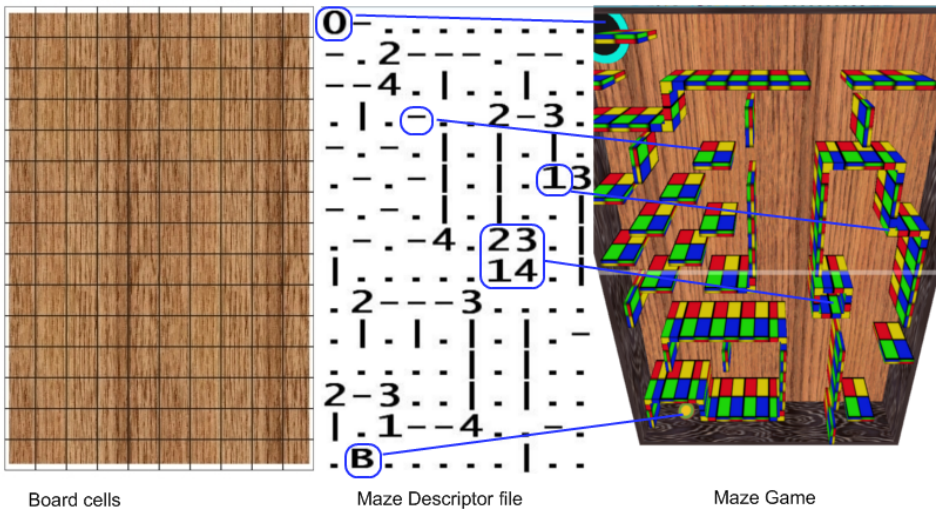


Figure 5.12: Maze Board subdivision, Maze Descriptor file, Maze Game

combination of all this objects with their respective orientations forms the entire maze. The use of a text file as a maze descriptor improves system loading speed because of its tiny size. Moreover it improves software maintenance because easy to modify on the fly without any internal code modifications.

5.3.4 Physical Simulation

The game maze is entirely physics based game. In order to simulate a realistic physical behaviour a physics engine has been used for the Maze game implementation. A physics engine is a software component that provides quite realistic physical simulation such as rigid bodies dynamics, soft body dynamics, fluid dynamics and mainly collision detections and responses. Physical needs for the game Maze were the implementations of rigid body dynamics, mainly spheres (for the ball) and boxes (for the walls and Boards Margins), gravity force, friction and kinematic laws to describe motions. Gravity, friction and kinematics were applied to the balls to simulate their behaviours. The Ball in fact needs gravity to fall on the board and kinematics to roll on it depending of its orientation. Collision detection is one of the most important

physical aspects of the game. The game concept is based on the balls collisions with board and walls. The physics engine is separate environment hence every object rendered on the 3D scene, if suppose to own physical properties, has to be replicated in the physic engine as a simplified object. Once all the objects are correctly replicated inside the physics engine for every frame, one step of the physic simulation is executed. Information about the board and wall rotations (which depend on users' inputs) are passed to the physics engine before executing its update step. Ball orientations and positions are instead read from the physics engine after its frame update; in order to obtain the correct positions and orientations, these values are read after the collisions detection and the relative impulse computations. One of the biggest challenges using physics engine is the synchronization among graphics objects and physical object inside the physics environment; 3D models and physical models may not have the same spatial and geometrical properties. One of most significant example in the Maze game is the object *Maze Margins*. The Maze Margins 3D model is a single geometrical object with its centre set at the 3D coordinates origin in the scene. In order to rotate the Maze margins following the board orientation, the *TotalRotation* matrix is applied as transform to the Maze Margins object. The Maze Margins object cannot be replicated inside the physics environment as a single object because of its concavity. In physics is much easier to cope with convex objects rather than concave ones; for this reason the geometry of the Maze Margins is decomposed in four adjacent boxes. These four boxes, in order to follow the board orientation, they need the *TotalRotation* matrix and translation displacement matrices to be correctly translated in the 3D space. Taking track of these representations difference between 3D models and physical models is quite a challenge even because the physic engine is not a visual environment.

5.4 Used Technology

For the final demo several software and piece of technology have been implied. The projector implied is the Macrovision SHOWWX laser pico projector; this pico-projector present very small sizes and a very bright image. The camera implied was Microsoft LifeCam Cinema Web camera because of its wide screen ratio and the superior image quality, in high definition. The programming language used was C++ for its execution speed and its flexibility in memory management. The graphics API used was Microsoft DirectX 9.0c for it flexibility in customization of graphics device. OpenCV library was used as image processing library implementing all the vision algorithms described. Bullet open source physics engine was used for the physical simulation. Autodesk 3D studio max was used to produce 3d model in the game Maze.

Chapter 6

Evaluation

An overall evaluation has been conducted to asses both the gestural interface goodness and the game impact on a representative user sample. Investigating on the gestural interface precision and robustness allow checking whether the interface is reliable or it needs further improvements; showing that an interface is reliable can bring more visibility to the gestural interface research field encouraging further research on this topic. Besides analyzing the overall game-gestural interface impact on final users provides an overview on the game potential success if released as a commercial product.

6.1 Technical Evaluation

The technical evaluation is aimed to bring numeric significant results useful for further consideration. Evaluating a user interface is a very important task because express in numbers how successful the application may be. In fact even the best good looking graphics game with the best plot, if it fails in recognizing user inputs it will be a total failure as a game. Moreover a technical evaluation is useful to identify limitations, weaknesses and development directions for future improvements. Four metrics have been developed to evaluate system *Robustness*, *Repeatability*, *Precision* and *Average*

frame rate. The *Repeatability* or *Test-retest Repeatability* (**REP**) expresses how good the system recognizes same hand gesture inputs. A same hand gesture is repeated for several times (**Nges**); also the number of times the system wrongly detects the inputs is counted. The *Repeatability* is calculated as eq. (6.1). The *Precision* (**P**) expresses how good the system recognize different hand gestures; several hand gestures are executed (**Inp**); also the number of times the system wrongly detects the inputs is counted (**Imiss**). The *Precision* is calculated as eq. (6.2). The *Robustness* (**ROB**) is the property of the system not to be affected by noise. It calculates the number of times in which the system is affected by noise (**Nnoise**) within a finite interval of minutes; the **Nnoise** refers to a discrete scale (table 6.1) which expresses how robust the system is to noise. The last metric is the *Average frame rate* of the implemented method indicated as **AVGFps**. The **AVGFps** calculates how many milliseconds in average the system takes to execute an entire vision pipeline cycle.

$$REP = \frac{Nges}{Nges + Imiss} \tag{6.1}$$

$$P = \frac{Inp}{Inp + Imiss} \tag{6.2}$$

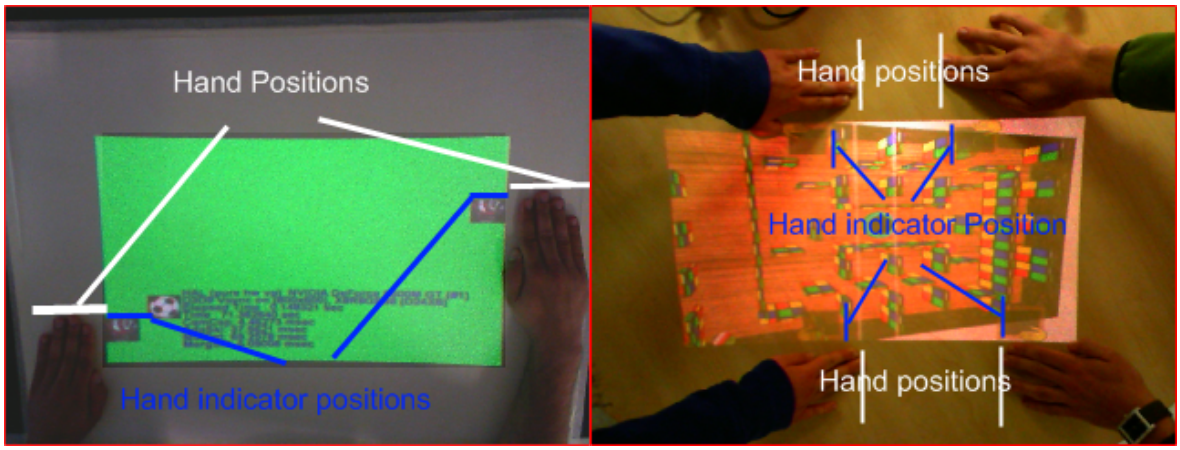
The *Repeatability* and *Precision* are obtained by a human executing the allowed hand

Nnoise	**Robustness**
$= 0$	Very Robust
$0 <$ Nnoise ≤ 2	Robust
$2 <$ Nnoise ≤ 4	Sensitive to Noise
> 4	Weak

Table 6.1: Robustness scale

gestures and checking visually when the system misses them. The visual survey is

possible through the hand indicators implemented in the game. The hand indicators follow the hand tips while hands move (Fig. 6.1a and 6.1b). If after the execution of



(a) Hand indicators Pong Game (b) Hand indicators Maze Game

Figure 6.1: Hands Indicators

a hand gesture, the hand indicator does not follow precisely the hand tip or it does not stop precisely where the hand tip stops, the hand gesture is considered missed. The *Robustness* is also a human survey; the system is left running without inputs for a finite time interval; the human detects if accidental inputs affect the system due to external noise. Accidental input can be recognized by the movement of the hand indicators that for a very robust system are suppose to stay still during the whole experiment duration. The *Average frame rate* is instead calculated by the system itself. The system has been designed in components; for each component execution there is a time consumption record; given the number of elapsed frames it is possible to calculate the average time consumption for each component. The *Average frame rate* is printed onto the projected game area setting up an option in debug mode. The four metrics have been registered for both the prototype and the final demo; for the *Repeatability* and *Precision* metric ten gesture were executed each. The *Robustness* has been registered over and interval of five minutes. Evaluation results for both Prototype

and final demo are reported in table 6.2. Results highlight noteworthy improvements

	Prototype	**Final Demo**
Repeatability	(10 /(10 + 5) = 0.66	10 /(10 + 1) = 0.90
Robustness	4 ⇒ Sensitive To Noise	0 ⇒ Very Robust
Precision	10 /(10 + 4) = 0.71	10 /(10 + 2) = 0.83
Average frame rate	95.2 mSec	57 mSec

Table 6.2: Experiment technical Result

in the final demo. The prototype *Repeatability* and *Precision* were somewhat sufficient while the *Robustness* was very low. The final demo instead demonstrates an optimal *Repeatability* and a good *Precision*. The *Average frame rate* in the final demo has been also halved thanks to a multi-threading system. The system also results very robust. Improvements and simplifications in the pipeline have led to these positive results in the final demo.

6.2 User Impact Evaluation

Submitting the gestural interface to users opinions and evaluations allow estimating the impact of such a system if commercialized. Ten subjects were asked to sit and play the game; they were asked to play the single player mode and then the multi player mode. After having played, users were asked to fill up an anonymous questionnaire about the playing experience (See Appendix questionnaire). Data have been analyzed anonymously and user signed a voluntary agreement. Every question in the

Subject	Concept	Projector	Hands	Gameplay	Success
1	4	3	4	3	4
2	4	4	3	3	3
3	4	5	5	3	4
4	4	3	5	4	4
5	5	4	5	5	4
6	4	4	4	4	4
7	5	5	5	4	5
8	5	4	4	4	3
9	4	3	4	5	3
10	4	5	4	4	4
Total	43	40	43	39	38
Mean	4.3	4	4.3	3.9	3.8
Standard deviation	0.48	0.81	0.67	0.73	0.63
Mode	4	4	4	4	4

Table 6.3: Users' answers

questionnaire had an answer in five Likert scale. The Likert scale is common scale used in questionnaire and most widely used in survey researches. Likert scaling is a bipolar scaling method measuring either positive or negative answers. Results to the questionnaire test are reported in the Table 6.3 and in Fig. 6.2. The questionnaire results reflect a positive impact of the gestural interface and the game. Overwhelming is the average answer results for the game concept; participants widely appreciated the Maze concept commenting that the game was pleasant and amusing to play. Besides users were astonished of being able to play the game just sliding their hands; in fact in average they express that the gameplay was easy to learn. Good was the answers to the use of the projector instead of a classic display. Fairly positive in the end was the answer to the success of this kind of interactive entertainment applications; reasons why the success answer was not that high are to set on the environment constraints; in effect the environment constraints reduce the flexibility and mobility of the appli-

cations consequently its success; eliminating environment constraints the application will increase in its visibility and flexibility being easily accepted by the final users. Noteworthy constructive critiques were raised about the maze gameplay when played multiplayer. Two players commanding the same board can reach easily a deadlock, which sometimes is hard to overcome because one of the two players has to give in the board position momentarily; avoiding deadlock in the game suggests also future improvements for the gameplay.

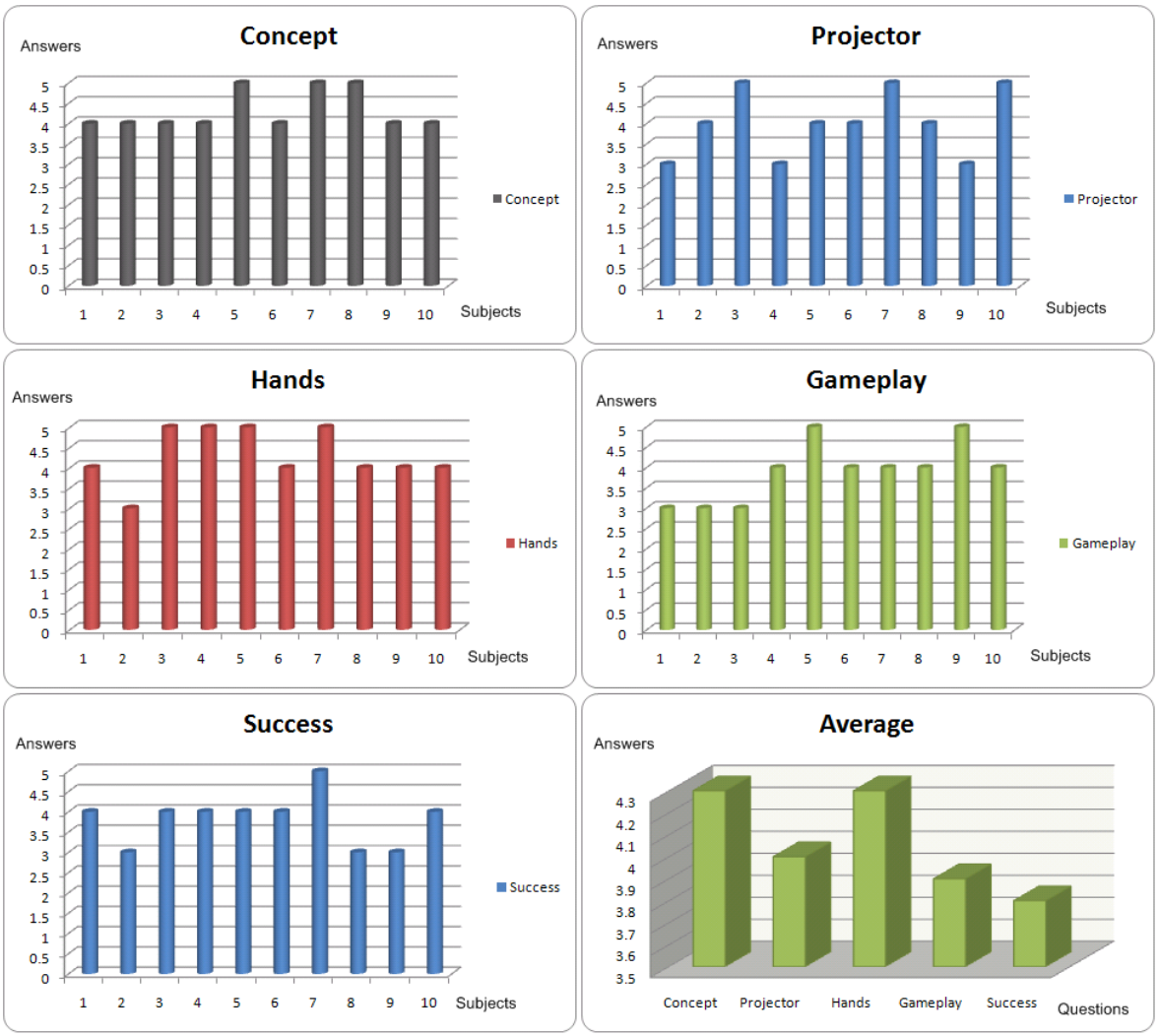


Figure 6.2: Subjects average

Chapter 7

Conclusions

In the end of this research work several interesting points have been brought to light. The research started with the investigation of technology and techniques and it concluded with their both technical and social evaluations. The technology investigation has shown that working with a coupled projector-camera has its advantages and disadvantages. Pico-projectors are small, tiny and powerful piece of technology that can be substitute to a display if the image quality is not a requirement; the projector image can be scaled depending how far the projector is from the projection surface. Projectors are so not constrained in sizes but their image brightness is inversely proportional to the projector distance from the projection surface. Moreover the projected image is sum of the image rendered on the projector and the projection surface material colours; cluttered surface colours reduce the projected image brightness and introduce surface colour noise in the projected image. Furthermore the projector image results clear to human eye but when it comes to use a camera to see the projector device image update rates should be taken into account. In fact during the implementation, the unsynchronized frame rate between camera and projector has been experienced. Usually camera frame rates are faster that projector ones which leads to capture the projector image while it is still being rendered, namely not complete. The average of

consecutive frames was the solution to this subtle issue. This simple solution is feasible in context where the camera image does not have to contain a clear and instantaneous image of the projected image. Lighting conditions in a coupled projector-camera technology have very different benefits; in effect their benefits are inversely proportional; in darker lighting conditions the projector image brightness increases while the camera image quality decreases due to noise. On the contrary with brighter lighting conditions the camera image quality improves while the projected image contrast decreases. A trick to improve camera image quality is tweaking the camera exposure depending on the environment light conditions. The exposure in fact commands the amount of light that passes through the camera pinhole. Investigation on segmentation techniques has shown that complex pipelines are less stable that simpler ones; every further steps introduce some errors due to techniques drawbacks. The Skin detection technique widely used in most of background hand recognition and detection applications has revealed not to be the best technique to segment hands in the context of this gestural interface. Skin detection even if adaptive, results highly light dependent; with lighting condition changes skin segmentation chances as well. Skin detection in fact is based on pixel colours which change rapidly with light. The attempt to minimize the skin detection instability merging skin segmentation with motion detection has brought to light other subtle issues: the moving shadows. A segmentation based on the moving areas takes into account any moving areas included the hand and arms moving shadows. Shadows in fact may represent either large or small area depending on the light direction respect to the hands; large shadows areas alter the hand segmentation including areas that are not hands. Moreover the motion detection introduces the moving hand assumption not being able to recognize stationary hands; this assumption results more a limitation in the gameplay which may require static inputs. The introduction of the adaptive background subtraction as main hand segmentation techniques has improved the stability, robustness and accuracy of the system as well as its average frame rate

thanks to multi thread programming. Stationary hand detection is also possible with background subtraction techniques enriching the precision of the input recognition.

Physics realistic games along with good inputs recognitions improve the users' amusement and increase the virtual reality users' immersion. The projector and the gestural interface help in improving the augmented reality feeling removing the display medium that highlights the separation between the user and the virtual reality rendered. The projector instead improves the visual link between user and the virtual game that becomes part of the user world being projected on a material touchable physical surface. The multiplayer game mode introduced an innovative aspect in a casual game context commonly only single player. A casual game should demonstrate fun in its users since the beginning to be successful; if the fun is multiplied by two players playing one against each other the fun raises.

The overall gestural interface with a projected display game impact on the interviewed users resulted very positive. Users liked the maze concept and playing it with their hands; moreover they believe that interactive application like this can gain success in the close future.

In conclusion the gestural interface with a projector display games lies on the "controller less" direction where all the new technologies are aiming. This research has demonstrated that such applications are feasible and their impact is positive on the users. Further research is suggested to improve gameplay and reduce environment and technology constraints.

7.1 Future Works

Future works for this project should be concentrating on the limitations that the system itself presents. Further research should be focus on minimizing the environment constraints. The projector-camera being fixed in its position represents a strict environ-

ment constraint; reasons why the projector camera cannot be moved from its original position are several. The projector has to be as orthogonal as possible to the projection surface otherwise the Keystone distortion will occur. The keystone distortion has been widely studied but to straighten the distorted projected image, the orthogonal distance between the projector and the surface, the tilt angle that differs from the orthogonal projector position need to be known. An improvement to the gestural interface would be to detect these two parameters through the camera. The amount of distortion could be evaluated projecting a grand truth image and capturing it with the camera to check how the distortion modified it; for examples using the same check board image used usually for the camera calibration, running a corner detection on that undistorted image, corners are suppose to lie on straight lines; if the projected check board image results distorted running a corner detection from a camera frame the distortion can be evaluated and corrected. In order to calculate the distortion precisely through the camera, frame rates between and camera need to be synchronized. An improvement to solve this issue would be to synchronize their update clock directly on their respective hardware. This method would assure a precise synchronized frame update clock. Further research would be spent on the improvement of hand recognition. Hand recognition in this research has been rejected because of a difficult separation of the gesture feature clusters. Researching new feature extraction technique could improve the separation between the gesture feature clusters. Adding actual gesture recognition in fact could enrich the gameplay adding new action available to the user to do during the game. One of the major constructive user critiques was that during the multiplayer mode, deadlocks in the gameplay could be reached. An improvement to this issue could be using the recognition of a special gesture (eg. a close fist) to execute a special action in the game to avoid deadlocks. An example could be the action of inhibiting the opponent from commanding the board for a small amount of seconds after collecting a certain number of points. Gesture recognition with the projected area would be one

of the biggest improvement and future work; in effect being able to detected gesture immerse in the projector light beam would allow new interesting way to interact with a game and consequently new interesting possible games. Dragging, pushing, pointing and pulling hand gestures being recognized within the projected image will allow the players to interact directly with the game objects augmenting the virtual reality experience. One of the biggest future work would the porting of the whole system on mobile phone to create a sort of "pocket entertainment"; In fact several prototype of mobile phones with embedded Pico-projector have already been released; one of the major assumption in that case should be that the projector lens and the camera one need to have the same physical orientation.

Appendix Questionnaire

Original Date:

GAME EVALUATION QUESTIONNAIRE

All questions contained in this questionnaire are strictly confidential
and will become part of anonymous user trial research.

How do you like the concept of the game?				
☐ I do not like it	☐ A little	☐ Interesting	☐ I like It	☐ I really like It
How do you like the projector approach instead of a classic display?				
☐ Terrible	☐ Poor	☐ Interesting	☐ I like It	☐ Impressive
How do you like using your hands to play the game?				
☐ I do not like it	☐ A little	☐ Interesting	☐ I like It	☐ I really like It
How intuitive did you find the gameplay?				
☐ Terrible	☐ A little	☐ Easy to learn	☐ Intuitive	☐ Natural
How successful do you think a gesture game with a projector would be?				
☐ Unsuccessful	☐ Moderately successful	☐ Successful	☐ Very successful	☐ Massive success
Do you have any comment or future improvement to the game? Please write below.				

Bibliography

[1] Apple iphone. http://www.apple.com/iphone/.

[2] Finger tracking - jonny lee project wii. http://johnnylee.net/projects/wii/.

[3] Macrovision pico-projectors. http://www.microvision.com/pico_projector_displays/embedded.html.

[4] Microsoft surface. http://www.microsoft.com/surface/.

[5] Perceptive pixel. http://www.perceptivepixel.com/.

[6] Project natal. http://www.xbox.com/en-US/live/projectnatal/.

[7] Crystal Muang Ahmed Elgammal and Dunxu Hu. Skin detection - a short tutorial. *Department of Computer Science, Rutgers University, Piscataway, NJ, 08902, USA*, 2009.

[8] Oliver Bimber. Spatial augmented reality. *SIGGRAPH 2007 Course 17 Notes Bauhaus University Bauhausstr. 11 99423 Weimar, Germany*, 2007.

[9] G. Bradski and A. Kaeheler. *Learning OpenCV, Computer Vision with the OpenCV library.* O'reillys, 2008.

[10] R. Mohor C. Schmid and C. Bauchkage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151172, 2000.

[11] Tudor Ioan Cerlinca, Stefan Gheorghe Pentiuc, Marius Cristian Cerlinca, and Radu Daniel Vatavu. Hand posture recognition for human-robot interaction. *MISI'07, Nagoya Japan.*

[12] Yen Ting Chen and Kuo Tsung Tseng. Developing a multiple-angle hand gesture recognition system for human machine interaction. *Mechanical and System research Laboratories, Industrial Technology Research Institute, Chuntung, Hsinchu 310, Taiwan*, 2007.

[13] Farhad Dadgostar. An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods. *Pattern Recognition Letters*, 27(12):1342–1352, 2006.

[14] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal refl ection. *Media Research Laboratory New York University 719 Broadway. New York, NY 10003*, 2005.

[15] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Ire transactions on information theory*, page 179, 1962.

[16] Byung-Kuk Junyeong Choi and Seo Jong-Il Park. Robust hand detection for augmented reality interface. *Department of Electronics and Computer Engineering Hanyang University, Seoul, Korea*, 2009.

[17] Michal Kawulok. Energy-based blob analysis fro improving precision of skin segmentation. *Institute of informatics, Silesian University of Technology, Gliwice, Poland*, 2010.

[18] Byungsung Lee and Junchul Chun. Manipulation of virtual objects in marker-less ar system by fingertip tracking and hand gesture recognition. *ICIS 2009, November 24-26, 2009 Seoul, Korea*, 2009.

[19] Darren Leigh and Paul Dietz. Diamond touch characteristics and capabilities. *Mitsubishi Electric Research Laboratories Cambridge, Massachusetts, USA*, 2001.

[20] Bruce D. Lukas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of the 7th international joint conference on Artificial intelligence*, 2(2):674–679, 1981.

[21] Frank D. Luna. *Introduction to 3D Game Programming with DirectX 9.0c A shader Approach*. WordWare Game and Graphics Library, 2006.

[22] V. Hlavac M. Sonka and R. Boyle. *Image processing, Analysis and Machine Vision*. Thomson, 2008.

[23] Shahzad Malik and Joe Laszlo. Visual touchpad: A two-handed gestural input device. *Department of Computer Science University of Toronto*, 2004.

[24] Daniel C. McFarlane and Steven M. Wilder. Interactive dirt: Increasing mobile work performance with a wearable projector-camera system. *Lockheed Martin Advanced Technology Laboratories, 3 Executive Campus, Cherry Hill, NJ 08002*, 2009.

[25] Pranav Mistry and Pattie Maes. Sixthsense: A wearable gestural interface. *SIGGRAPH Asia 2009, Yokohama, Japan, December, 2009*.

[26] Pranav Mistry, Pattie Maes, and Liyan Chang. Wuw - wear ur world - a wearable gestural interface. *CHI 2009, April 4 9, 2009, Boston, MA, USA*, 2009.

[27] David Molyneaux and Hans Gellersen. Projected interfaces: Enabling serendipitous interaction with smart tangible objects. *TEI 2009, February, 2009, Cambridge, UK.*, 2009.

[28] Nikola Pavei, Slobodan Ribari, and Benjamin Grad. Comparison of pca -, mdf -, and rd-lda - based feature extraction approaches for hand-based personal recognition. *International Conference on Computer Systems and Technologies - CompSysTech07*, 2007.

[29] Ronald A. Petrozzo and Stuart W. Singer. Cinema projection distortion. *The 141st SMPTE Technical Conference and Exhibition Displays for the Theater and Home Session*, 1999.

[30] Ken Schwaber. *Agile Project Management with SCRUM*. Microsoft, 2004.

[31] Abdesselam Bouzerdoum Son Lam Phung, , and Douglas Chai. Skin segmentation using color pixel classification: Analysis and comparison. *School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, NSW 2522, Australia.*, 2004.

[32] Alexandra Stefan, Vassilis Athitsos, Jonathan Alon, and Stan Sclaroff. Translation and scaleinvariant gesture recognition in complex scenes. *Computer Science Department, Boston University, Computer Science and Engineering Department, University of Texas at Arlington*, 2009.

[33] Liu Yun and Zhang Peng. An automatic hand gesture recognition system based on viola-jones ans svms. *College of information science ad technology, Qingdao*, 2009.