Spatial Augmented Reality on Mobile Robots

Nuno Gonçalo Trindade Lopes da Costa

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Artur Miguel do Amaral Arsénio

Examination Committee

Chairperson: Prof. João Paulo Marques da Silva
Supervisor: Prof. Artur Miguel do Amaral Arsénio
Member of the Committee: Prof. José António da Cruz Pinto Gaspar

November 2014
In memory of my father...
Acknowledgments

I would like to begin by thanking my supervisor, Dr. Artur Arsénio for all the support, invaluable advice, guidance and interest through all the conversations we had during the course of this thesis which were essential for its development and conclusion. I would also like to express my gratitude to all others who, directly or indirectly, were involved in my academic journey and upcoming for their patience, knowledge and wisdom.

To my colleagues at YDreens that soon became friends, with their invaluable support and help that cheered me with so many good moments! Special thanks to Filipe Tavares for his help with his work on the camera localization system.

Last but not least, I would like to thank my family and friends, in particular my mother for all the love and support throughout my life. To my father, wherever you are, my promise is kept.
Resumo

Avanços na área de Realidade Aumentada levou-nos para além do típico sistema acoplado ao utilizador e permitiu a criação de um novo ramo: Realidade Aumentada Espacial, criando áreas aplicacionais outrora inexistentes. Este novo ramo é uma área de computação gráfica/realidade virtual emergente que permite não só introduzir objectos virtuais sob ambientes reais recorrendo a projectores digitais mas igualmente impôr restrições virtuais ao conteúdo a ser projectado baseadas nas restrições físicas existentes no mundo real.

Esta tese investiga o uso de um sistema de Realidade Aumentada Espacial em robôs presentes na ala de pediatria do Instituto Português de Oncologia em Lisboa, fazendo parte de um projecto europeu em curso denominado MOnarCH (Multi-Robot Cognitive Systems Operating in Hospitals). Os sistemas de Realidade Aumentada Espacial actualmente existentes são maioritariamente estáticos, ao adicionar esta capacidade a uma plataforma com mobilidade surge uma nova limitação: de forma a correctamente poder visualizar objectos virtuais independentemente do ponto de vista do projector ou da posição e/ou orientação da superfície de projeção é necessário aplicar uma pre-distorção à imagem a ser projectada no mundo real.

A aplicação desenvolvida terá como input dados de calibração e localização do projector, localização de superfície(s) de projeção e skeleton tracking obtido a partir do sensor de profundidade de uma câmara. O output da aplicação será a projeção de uma aplicação interactiva de Realidade Aumentada sob o ambiente real, devidamente compensada de forma a remover a distorção decorrente deste tipo de situações.

Abstract

Novel approaches have taken Augmented Reality (AR) beyond traditional body-worn or hand-held displays, leading to the creation of a new branch of AR: Spatial Augmented Reality (SAR) providing additional application areas. SAR is a rapidly emerging field that uses digital projectors to render virtual objects onto 3D objects in the real space.

In this work it is investigated the use of SAR-enabled robots to be present in the pediatric ward of Instituto Português de Oncologia de Lisboa (Portuguese Oncological Institute of Lisbon), part of an ongoing project called MOnarCH (Multi-Robot Cognitive Systems Operating in Hospitals). When adding SAR capabilities to a mobile platform such as a robot, a new limitation rises: in order to project the augmenting virtual objects from any viewpoint without distortions, the final image should be pre-distorted based on the position and orientation of the real world objects being projected upon in relation to the projector mounted on the robot. The projection surface geometry information will also be used to enforce virtual restrictions over the projected content based on the physical restrictions of the real world objects being projected upon.

The application developed will have as input projector calibration and positioning information, localization of the intended projection surface(s) location and skeleton tracking supplied by a camera with a depth sensor. The output will be the projection of undistorted Augmented Reality content onto real world planar surface(s).

Keywords: Spatial Augmented Reality, Augmented Reality, Projection, Distortion, Mobile Robots, Human-Robot Interaction.
## Contents

Acknowledgments ................................................................. v
Resumo ..................................................................................... vii
Abstract ..................................................................................... ix
List of Tables ............................................................................... xv
List of Figures .............................................................................. xx
Nomenclature .............................................................................. xxi
Glossary ...................................................................................... xxv

1 Introduction ................................. 1
   1.1 Motivation ................................................................. 1
   1.2 Objectives ................................................................. 2
   1.3 Contribution ............................................................ 2
   1.4 Document Outline .................................................... 4

2 State of the Art ................................. 5
   2.1 Augmented Reality ...................................................... 5
      2.1.1 Importance and Applications .............................. 6
      2.1.2 Challenges ......................................................... 7
      2.1.3 Concepts ........................................................... 9
   2.2 Spatial Augmented Reality ........................................ 16
      2.2.1 Introduction ....................................................... 16
      2.2.2 Motivation ......................................................... 17
      2.2.3 Related work ..................................................... 18
      2.2.4 Methods .......................................................... 19
      2.2.5 SAR Limitations ............................................... 20
      2.2.6 The Future ....................................................... 21

3 Camera ......................................................... 22
   3.1 Introduction ............................................................. 22
   3.2 Homogeneous Coordinates ...................................... 22
   3.3 Pinhole camera ......................................................... 24
   3.4 Synthetic Camera ..................................................... 25
# Camera-Projector Calibration

## 4.1 Camera calibration

## 4.2 Determining Camera Position and Pose

## 4.3 Projector calibration

# Unity Platform

## 5.1 Introduction

## 5.2 Concepts

### 5.2.1 Assets, Game Objects and components

### 5.2.2 Scripts

### 5.2.3 Coordinate System

### 5.2.4 World Space vs Local Space

### 5.2.5 Vectors

### 5.2.6 Meshes, polygons, edges and vertices

### 5.2.7 Material, textures and shaders

## 5.3 Camera Model

### 5.3.1 Default: gluPerspective

### 5.3.2 Projector-based approach: glFrustum

# Implementation

## 6.1 Architecture

## 6.2 Homography

## 6.3 Scripts Overview

## 6.4 Game Object Overview

## 6.5 Source and Destination Points

## 6.6 Projection Surfaces

## 6.7 Interactive AR Content

## 6.8 Saving and Loading States

## 6.9 Matching Virtual Camera position with Projector’s

# Evaluation

## 7.1 Hardware Setup

## 7.2 Scenario

## 7.3 Calibration Process

### 7.3.1 Camera

### 7.3.2 Projector

## 7.4 AR Projection

### 7.4.1 Scenario 1

### 7.4.2 Scenario 2

## 7.5 Air Hockey game
7.6 Discussion ................................................................. 64

8 Conclusions ............................................................... 66
8.1 Limitations .............................................................. 67
8.2 Future Work ............................................................. 67

Bibliography ................................................................. 75
List of Tables

7.1 Hardware specification ................................................................. 56
7.2 Camera calibration results ............................................................... 59
7.3 Projector calibration results ............................................................ 60
7.4 Projection results for the 47°scenario. .............................................. 62
7.5 Projection results for the 60°scenario. .............................................. 63
List of Figures

1.1 (a) Common scenario, where traditional projectors are orthogonal and create rectangular images; (b) The work on this thesis enables the projection onto oblique surfaces while having the projector in a orthogonal position with respect to a known wall. If the mobile platform on which the projector is mounted has a localization system (used by some robots, where data from motion sensors is used to estimate change in position over time), the projector can then move in four directions (left, right, forward and backward) while having the image compensated according to the new projector position; (c) It is possible extend the previous scenario and project from any given angle however this is only possible for pre-determined positions that have been previously (manually) calibrated, as such its usefulness is quite limited.  

2.1 A 3D model of a baseball player is displayed on top of a baseball card containing the picture of that very same player. (Source: Design Aerobics 2011, tech gadgets course sample lesson)  

2.2 Reality-Virtuality Continuum [Milgram et al., 1994], showing the two sides of mixed reality: Augmented Reality and Augmented Virtuality.  

2.3 A comparison of the proposed definition with the "ideal" definition of AR.  

2.4 Categorization of Challenges in Augmented Reality.  

2.5 Typical AR System Framework modules with their respective description.  

2.6 Classification of AR Tracking.  

2.7 The evolution of markers in pattern/image recognition technology related to AR applications.  

2.8 Visual display techniques and positioning.  

2.9 The underlying physical model of the Taj Mahal (top left) and the same model enhanced with Shader Lamps (bottom).  

3.1 3D view of a pinhole camera. A solid box containing a small pinhole in the front side. The origin of the coordinate system is set the pinhole, with all 3 imaginary axis represented.
3.2 Side-view of a pinhole camera box: rays of light from the exterior go through the aperture and are projected on the image plane, represented by the back-side of the pinhole box. Due to the aperture being so tiny, only one ray from a single point can go through the aperture. In this particular example, a single ray of light from the top of the house lands on the back-side of the box, designated projective or image plane.

3.3 A different perspective of the pinhole camera model where two distinct planes are represented: focal plane, corresponding to the front-side of the pinhole box where the aperture is located and the image plane, corresponding to the back-side of the pinhole box where the image is projected.

3.4 A point \( P = (X, Y, Z) \) is projected onto the image plane by the ray passing through the center of projection resulting in a point \( p = (x, y, f) \). In this variant of pinhole camera model, the image plane was moved from the back to the front of the center of projection. While the resulting geometry of point projection behind this change is equivalent, it simplifies the process.

4.1 Chessboard pattern with its features already calculated. Each feature is located in the interior intersections where two edges come together and then split off at opposing angles. Also, the origin is set to the bottom left corner but it could just as easily be set to any of other 3 corners.

4.2 Converting from object to camera coordinate systems: the point \( P \) on that object is seen as point \( p \) on the image plane. The relation between these two points is expressed by applying a rotation matrix \( R \) and a translation vector \( t \) to the point \( P \).

4.3 Steps taken to find the projector position in relation to a fiduciary marker coordinate system represented by step 5. The point \( P_m \) on the fiduciary marker is seen as point \( p_c \) on the camera image plane, which in turn is seen as point \( p_p \) on the projector "image plane". The relation between each coordinate system is achieved by applying a rotation matrix and a translation vector to each point.

4.4 Modified version of ArUco: the camera position and pose (bottom left) are determined with respect to a marker, placed on the wall. The entire work on this thesis is based on this very same referential.

5.1 Each object in Unity can be represented using two space coordinate system: world and local.

5.2 On the left: a 2D perspective where "field of view Y" represents the angle formed by the top and bottom of the frustum in the YZ plane. On the right: a 3D visualization of the actual frustum where the aspect ratio is represented by \( \frac{\text{width}}{\text{height}} \).

5.3 Left and right are measured along the X axis, top and bottom along the Y axis. The blue and green plane correspond to the near and far parameters respectively.

5.4 Correspondence between focal plane and image plane.
5.5 Normalized device coordinate space, represented by a cube with the front-side and back-side represented by near and far planes respectively. The origin of this coordinate system is set in the cube’s geometric center. Each corner corresponds to the glFrustum method parameters for the near and far planes.

5.6 Top-view of the frustum, showing the correspondence between the focal and near planes.

6.1 Architecture for the proposed solution. Besides the main application developed in Unity, a second application was also developed using WPF to simulate a robot’s position (and orientation) stream of information onto Unity. The line in dash means a connection not implemented due to the lack of a fully functional robot at the time this thesis’ work was developed and tested.

6.2 Point mapping from one point in a plane onto another point in a different plane. The projection also maps lines to lines as may be seen by considering a plane through the projection centre which intersects with the two planes $\beta$ and $\beta'$. Since lines are mapped to lines, central projection is a projectivity and may be represented by a linear mapping of homogeneous coordinates $x' = Hx$.

6.3 Hierarchy representation in the form of parent-child of a Projection Surface game object. Each element represents a unique game object present in Unity scene.

6.4 Example of the positioning of both source and destination points in a Unity scene. The position of any destination point can be manipulated at any time. By doing so, a new homography is automatically updated and its result can be instantly visualized. At any point each destination point is “attached” to each corner of the image plane.

6.5 Vertex shader transformation used for each projection surface. The diagram annotates the transitions between each transform with the coordinate space used for vertex positions as the positions pass from one transform to the next. The resulting homography matrix is computed within a C# script and its value is passed onto directly the shader so that the transformation can be applied to the projection surface mesh.

6.6 Representation of the Air Hockey game. Each player controls a paddle with the objective of scoring points on the opponent’s goal.

6.7 The game starts using Kinects skeleton tracking feature. One or two players are tracked simultaneously (step 1). The corresponding joint of each player controls the respective paddle in the game (step 2). Camera 2 renders the texture containing the camera view (step 3) which is then applied to one of the projection surfaces and updated in runtime (step 4). The final step can now take place: project onto some real world surface (step 5).

6.8 XML file content containing two projection surface coordinates. At any time the user can save or load the coordinates through the use of a keybind (‘S’ key to save, ‘L’ to load). Each time the coordinates are loaded into Unity, every destination point game object position is automatically updated and its corresponding homography matrix calculated.
6.9 Basic application where the user can insert values representing the new position of the projector. This information is sent into Unity, updating its virtual camera position so that it matches the projector one, which in turn, triggers the update of all homography matrices of the projection surfaces present in the scene.

7.1 On the left, a visualization of the experiment scenario: an A4 sheet is attached to a cardboard, at a $\beta$ angle with respect to the projector. AR content is then projected onto that sheet, measurements then take place between each corner of the page and its corresponding corner of the projected content. This process is then repeated up to 11 times for each $\beta$ angle (the chosen angles were 47° and 60°). The coordinate system used by a fiduciary marker placed in a known position in the real world is used as the systems coordinate system. On the right: the actual experiment taking place.

7.2 Experimental process workflow.

7.3 On the left the results of camera calibration process: the camera is represented in red with its own referential (in blue) where each grid represents a chessboard pattern with unique position and pose with respect to the camera. On the right, one of the images used by the calibration application. Values are measured in mm.

7.4 On the left, a representation of the mean reprojection error per image (lower is better). On the right, the same reprojection error from a pixel point of view, ideally each dot needs to be as close to the origin (center of image) as possible.

7.5 On the left, the result of camera-projector calibration procedure displaying the projector position in relation with the camera coordinate system; the camera is represented in red, projector in green and each grid represents a chessboard pattern with unique position and pose with respect to both camera and projector. On the right, the actual procedure taking place. All values are measured in mm.

7.6 Each dot represents the reprojection error (from a pixel point of view) of a feature from the chessboard. The closer they are to origin (center of image), the smaller will be the error.

7.7 Projection results obtained with the projection surface at an angle of 47° with respect to the projector. Each marker $x$ represents the physical coordinate of a corner of the target projection surface (A4 sheet). The remaining markers represent measurements taken at different distances: 1.2m, 1.6m and 2.2m represented by blue star, green cross and red square respectively. The offset values are in cm.

7.8 Projection results obtained with the projection surface at an angle of 60° with respect to the projector. Each marker $x$ represents the physical coordinate of a corner of the target projection surface (A4 sheet). The remaining markers represent measurements taken at different distances: 1.2m, 1.6m and 2.2m represented by blue star, green cross and red square respectively. Values measured in cm.

7.9 The game, the mobile robot, and people playing the air hockey game as projected by the robot on a non-aligned plane of projection.
Nomenclature

Subscripts

\(x, y\) 2D Cartesian components.

\(x, y, w\) 2D Homogeneous components.

\(x, y, z\) 3D Cartesian components.

\(x, y, z, w\) 3D Homogeneous components.

Superscripts

\(T\) Transpose.
| **AR**  | Augmented Reality is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. |
| **DGPS** | Differential Global Positioning System is an enhancement to Global Positioning System that provides improved location accuracy, from the 15-meter nominal GPS accuracy to about 10 cm in case of the best implementations. |
| **DPI** | Dots per inch is a measure of spatial printing or video dot density, in particular the number of individual dots that can be placed in a line within the span of 1 inch (2.54 cm). |
| **GPS** | Global Positioning System is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. |
| **HMD** | Head-Mounted display or Helmet Mounted Display, is a display device, worn on the head or as part of a helmet, that has a small display optic in front of one or each eye. |
| **HRI** | Human Robot Interaction is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans. |
IDE  Integrated Development Environment is a software application that serves as a platform to computer programmers for software development.

IPOL  Instituto Português de Oncologia de Lisboa is a hospital located in Portugal that has as main focus the treatment patients affected by cancer.

ISAR  International Symposium on Augmented Reality is a former academic event dedicated to this research field. It is now known as ISMAR.

ISMAR  International Symposium on Mixed and Augmented Reality is the leading international academic conference in the field of Augmented Reality and Mixed Reality.

ISMAR  International Symposium on Augmented Reality is a former academic event dedicated to this research field. It is now known as ISMAR.

IWAR  International Workshop on Augmented Reality is an event where researchers report their latest concepts, progress, and results, and sessions where short positional statements trigger discussions about the future path of AR technology.

JIT  Just In time compilation is compilation done during execution of a program rather than prior to execution.

MOnarCH  Multi-Robot Cognitive Systems Operating in Hospitals is an European project that involves researchers from approximately ten European companies and research centres, whose goal is to develop and introduce a fleet of robots that collaborate with medical personnel and interact with patients.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAR</td>
<td>Spatial Augmented Reality (SAR) augments real world objects and scenes without the use of special displays such as monitors, head mounted displays or hand-held devices. SAR makes use of digital projectors to display graphical information onto physical objects. The key difference in SAR is that the display is separated from the users of the system.</td>
</tr>
<tr>
<td>VE</td>
<td>Virtual Environment is a computer-generated, three-dimensional representation of a setting in which the user of the technology perceives themselves to be and within which interaction takes place; also called virtual landscape, virtual space, virtual world.</td>
</tr>
<tr>
<td>WPF</td>
<td>Windows Presentation Foundation provides developers with a unified programming model for building rich Windows smart client user experiences that incorporate UI, media, and documents.</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Robotic applications for health care have skyrocketed over the past decade [Xing et al., 2013]. Besides great improvements in surgical robots, robotic solutions are also being applied to a much wider range of scenarios, such as patient care, human rehabilitation, hospital logistics, assisted living or social therapy, among others. Human–Robot Interaction (HRI) aims at comprehending, building, and evaluating robotic systems to be used by humans, or to cooperate with them, and that interact with humans in meaningful ways [Michael and Schultz, 2007]. Interaction per se requires a communication channel between robots and humans, such as voice (audio or speech synthesis and recognition), and gestures.

Pediatric care is a particular area of interest on the HRI domain, not only due to the young age of the target user population, but also due to the fact that it can be quite challenging to entertain sick children. Adding to that, moving robots from controlled environments like laboratories into a dynamic and demanding environment like a pediatric ward of a hospital presents unique technological challenges, ranging from autonomous navigation through the wards using a robot localization system, to human-like interactions with children or medical personnel. Additionally, the robots are supposed to inform and entertain children, projecting augmented reality information on the environment, with the dual role of providing information to children, as well as to interact with children for game playing and social entertainment. It also presents challenges from a sociological point of view since there are very few studies looking into relationships between humans and robots over long term\(^1\).

Several studies [Dautenhahn, 2000] have shown that interactions between patients and robots can be beneficial for the former, especially for certain type of patients. For instance, studies in the United Kingdom explored the possibility of using social robots with autistic children. In Europe, the EMOTE (Robotic Tutors for Empathy based Learning) research project\(^2\) is exploring the usage of social robots for teaching children. On the other hand, the robot Paro in Japan (a robot resembling a baby seal, with expressive eyes) was reportedly able to improve the mood of elderly people, and simultaneously

\(^1\)http://www.eurekalert.org/pub_releases/2013-03/ciuo-rot03ii13.php (accessed last time on September 22nd, 2014)

\(^2\)http://www.emote-project.eu/ (accessed last time on October 14th, 2014)
reduced stress not only to patients but also to their caregivers [Wada and Shibata, 2008]. This has been demonstrated more recently to treat some cases of depression suffered by the survivors of the devastating earthquake and tsunami in the northeast coast of Japan in March of 2011.

1.2 Objectives

This work is being developed on the scope of the MOnarCH project (Multi-Robot Cognitive Systems Operating in Hospitals) focused on introducing a fleet of social robots that will collaborate with medical personnel, interacting with children, who are patients in the pediatric ward of the hospital Instituto Português de Oncologia de Lisboa (IPOL). The objective of this project is to further improve the HCI capabilities, making a significant qualitative leap forward by using not one, but several robots of two distinct formats, each with its own unique capabilities complementing one another through a cooperative behavior, i.e. they communicate with each other in order to fulfill their objectives.

This work is also being developed for an Augmented Human Assistance environment, on the scope of a CMU-Portuguese collaborative research program aiming at patient rehabilitation after injury, allowing the execution of exercises on remote locations without requiring the patient to move into a clinical health provider.

The work developed in this thesis aims at providing projection mapping capability, also known as Spatial Augmented Reality (SAR) to mobile platforms like robots. This projection technology is used to turn objects, often irregularly shaped, into a display surface for video projection. Using specialized software, a two or three dimensional object is spatially mapped on the virtual program, mimicking the real environment which will be projected upon. The software can interact with a projector to fit any desired image onto the (planar) surface(s) of that object. The SAR system developed is based on three key hardware components:

- A Kinect depth camera used for human-robot interactions;
- A projector used to project Augmented Reality (AR), a cutting-edge technology that provides a digitally enhanced view of the real world;
- A second camera that serves two purposes: projection calibration as well as the robots localization system used to detect the presence of fiducial markers in the environment;

Both concepts, AR and SAR, will be thoroughly described in chapter 2 with their respective state of the art.

1.3 Contribution

The goal of this dissertation is further improve the existing HRI by adding to robots a unique capability: SAR. Like robots, this technology can exist on its own however by combining both it allows the creation
of unique interactive AR scenarios that otherwise would not be possible. The SAR system developed is divided in two stages, as such, this thesis has two main contributions:

- Development of an application using Unity game engine that allows the creation of planar surfaces that will be used to project AR content onto real world surfaces. An algorithm will be applied onto the texture of each surface so that the perspective distortion caused by its position and orientation in respect to the projector will be compensated, leading to a distortless image;

- Apply all the data output from the projector calibration process, namely extrinsic and intrinsic parameters, to Unity’s virtual camera, so that it emulates the behavior of a real world lens as close as possible;

One can consider the first contribution as a correction to the perspective distortion problem applied in a virtual world. The second contribution focus on applying real world constraints inherent to a digital projector lens (that come into play when projecting the virtual output of the developed application) onto the real world.

The end goal is to have a robot projecting AR on multiple surfaces, while being idle or moving, performing the necessary compensations for each surface being projected. This way, projected AR content will look undistorted, regardless of the robots position/orientations with respect to each surface.

Fig.1.1 illustrates a visual comparison between an everyday projection scenario and two case scenarios made possible with this thesis proposed solution.

![Figure 1.1](image.png)

Figure 1.1: (a) Common scenario, where traditional projectors are orthogonal and create rectangular images; (b) The work on this thesis enables the projection onto oblique surfaces while having the projector in a orthogonal position with respect to a known wall. If the mobile platform on which the projector is mounted has a localization system (used by some robots, where data from motion sensors is used to estimate change in position over time), the projector can then move in four directions (left, right, forward and backward) while having the image compensated according to the new projector position; (c) It is possible extend the previous scenario and project from any given angle however this is only possible for pre-determined positions that have been previously (manually) calibrated, as such its usefulness is quite limited.
### 1.4 Document Outline

The work reported by this document is organized throughout eight chapters, as follows. **Chapter 1**, in which this section is included, introduces the motivation for this thesis, its objectives and contributions. **Chapter 2** contains the state of the art more relevant for two key concepts addressed by this thesis: Augmented Reality and Spatial Augmented Reality. In **Chapter 3** two basic concepts underlying this thesis work are introduced: the homogeneous coordinates space and the core camera model. **Chapter 4** describes the camera and projector calibration procedures, applied with the main goal of determining the projector’s corresponding intrinsic and extrinsic parameters. **Chapter 5** introduces the core concepts of the Unity game engine, including a detailed explanation behind the changes made to its camera model. **Chapter 6** depicts the solution architecture, with a thoughtful description for each of its components. **Chapter 7** presents the experimental evaluation process for this work, quantifying the error margins caused by distinct error sources, namely camera and projector calibration, as well as the projection itself and projective distortion compensation. Finally, **Chapter 8** draws the main conclusions concerning the contributions of this work, as well as its limitations, and addresses future directions for research work.
Chapter 2

State of the Art

2.1 Augmented Reality

Augmented Reality (AR) is a technology through which the view of real world environment is augmented (or supplemented) by computer-generated elements or objects such as sound, video or graphics that appear to coexist in the same place as the real world. It is a promising user-interface that has the potential of changing the way that users view the real world and manipulate 3D objects. An example of AR can be seen in Fig. 2.1.

![Augmented Reality Example](image)

Figure 2.1: A 3D model of a baseball player is displayed on top of a baseball card containing the picture of that very same player. (Source: Design Aerobics 2011, tech gadgets course sample lesson)

With recent computer vision and mobile device technology advances, namely computational power, storage capacity and cloud service availability, AR has become a tool available to the masses. As shown in figure 2.2, Augmented Reality lies between the virtual world and the real one.

![Reality-Virtuality Continuum](image)

Figure 2.2: Reality-Virtuality Continuum [Milgram et al., 1994], showing the two sides of mixed reality: Augmented Reality and Augmented Virtuality.

The Reality-Virtuality Continuum described by Milgram [Milgram et al., 1994] shown in Fig. 2.2 de-
picts the range from the real (i.e. Physical) environment to the immersive virtual one that completely immerses a user inside a synthetic environment where the user can neither see nor interact with the real world. Between these two extremes there is a blend notion between the real and virtual, hence the term Mixed Reality, which includes Augmented Virtuality where most of the input (often the background) is computer-generated and real world objects are added to the virtual scene, in contrast with Augmented Reality where virtual objects are merged or added to the real world. Lester Madden’s [Madden, 2011] description of AR technology lies on five features:

- Combines the real world with computer graphics;
- Allows interaction with objects in real-time;
- Tracks objects in real-time;
- Provides recognition of images or objects;
- Enable real-time context or data;

Unfortunately this definition does not capture all the essential AR features, it is merely a proposed definition but not the ideal one.

As shown in figure 2.3, both AR definitions are not fully complete. For example, a location-based AR which is one of the main types of AR does not include all of mentioned features since it does not track objects nor it is based on recognition [Geroimenko, 2012].

The concept of AR applies to all human senses, i.e. the five natural powers of sight, hearing, feeling, taste and smell, being visual enhancement the most common type of AR and the one that will be focused throughout in this paper.

### 2.1.1 Importance and Applications

So why is AR important? At a first glimpse, for the average computer user, combining real and virtual objects in 3-dimension (3D) might seem little more than an entertainment experience, however when you start thinking in its potential in several areas it suddenly provides a whole new way of how we can see and interact with the environment around us. Some of these areas [Azuma, 1997] are:

- Medical visualization where doctors can use AR as a visualization and training aid tool for surgery. It is also possible to collect 3D datasets of a patient in real time using non-invasive sensors like Magnetic Resonance Imaging (MRI) or Computed Tomography scans (CT). These datasets can
be rendered and combined in real time providing an internal view of the real patient. One could compare this type of procedure as having an “X-ray vision” of the patient’s interior body;

- Manufacturing or repairing complex machinery while interpreting instruction not from a standard manual with text and pictures but rather from 3D drawings superimposed upon the real world equipment showing step-by-step what needs to be done and how to do it or in an even more intuitive way, using animations. There are several research projects, along with its prototypes, for example Feiner’s group at Columbia built a laser printer maintenance application [Feiner et al., 1993b];

- Annotation and visualization, where AR can be used to annotate objects and environments with some desired public or private information. This sort of AR is frequently used in museums and libraries where visitors can have information related to the objects they are looking at displayed on a hand-held or mobile phone device. Researchers at Columbia demonstrated this by displaying windows from a standard user interface onto specific locations in the real world or simply attaching them to specific objects serving as reminders [Feiner et al., 1993a];

- Robot path planning, since operating a robot from far away can become problematic due to latency issues. One possible solution for this problem is instead operate a local virtual version of the robot, where the user defines and tests the desired path and then sends over to the robot the instructions to execute it leading, to some extent, to a seamlessly experience. The ARGOS system has demonstrated that the use of stereoscopic AR is an easier and more accurate way for robot path planning compared to the traditional monoscopic interfaces [Drascic, 1993, Milgram et al., 1993];

- Entertainment, allowing the introduction of objects that do not exist in the real world or simply as a way to reduce production costs since being able to create and reuse virtual sets can be potentially cheaper than having to constantly build new physical sets from scratch. The ALIVE project from the MIT Media Lab goes one step further by populating the environment with intelligent virtual creatures that respond to user actions [Maes, 1995];

- Military aircraft, where AR has become an essential tool for aircraft and helicopters pilots by using Head-Up displays (HUDs) and Helmed-Mounted Sights (HMS) which superimpose vector graphics upon the pilots view of the real world displaying all sort of relevant information;

2.1.2 Challenges

Even though the concept of AR has been around for few decades and a considerable amount of work has been made in the area it is still far from being a mature technology. There are numerous challenges of different types to be surpassed, which are displayed in figure 2.4 and described in the following sections.

Performance Real time processing can be a major bottleneck of any AR application especially if its a mobile one [Rabbi and Ullah, 2013]. Even the recognition of simple markers can be computationally
very expensive [Wagner, 2009]. This latency issue can slow down the performance of a AR application to a point where it can significantly affect the user experience.

Alignment  One of the main objectives when providing the user an AR experience is assuring that the virtual objects are properly placed in the real world, whether they are being superimposed or just added. That process is known as alignment and an incorrect alignment would translate into misplaced information in the real world which could have major implications in areas like medical applications. Alignment challenges include registration problems, which are the most basic problem in AR [Azuma, 1997]. Registration is the proper alignment of virtual objects to the real world objects [Azuma, 1997, Hoff et al., 1996]. Outdoor AR tracking can also be an issue since even the smallest tracking error can cause a clear misalignment between the virtual and real objects [Wang and Dunston, 2007].

Interaction  This particular challenge refers to the interaction of users with virtual and real objects at the same time. There are several interfaces that can be used in this interaction, such as acoustic, haptic, gaze or text-based. Ulhaas and Schmalstieg introduced a finger tracker which is based on a special glove with retro-reflective markers [Ulhaas and Schmalstieg, 2001]. Interaction Techniques and User Interfaces are still problems which need to be solved [Zhou et al., 2008].

Mobility  This challenge is a major concern when using mobile AR systems. Ideally the system should be small and light so that it can be used anywhere. The first mobile AR systems were not a viable option for a daily use due to their size and weight which made them quite cumbersome for the user. With recent technological advances in mobile devices regarding the computational power and storage capacity this type of challenge has been solved for some AR applications, like AR browsers running on smartphones. The ideal AR system will be portable outside a controlled environment [Azuma, 1997].

Visualization  There are various potential visualization issues such as display (HMD based or monitor-based), contrast, resolution, brightness and field of view. The illumination of the virtual and real world object is required to be the same [Fournier, 1994, Drettakis et al., 1997]. Another potential visualization issue is occlusion, i.e. the process that determines which surface or parts are not visible from a certain view-point [Wang and Dunston, 2007, Fuhrmann et al., 1999]. A semi-automatic solution for this problem has already been presented by Lepetit and Berger [Lepetit and Berger, 2000].

Besides these challenges and as AR evolves, several areas of concern will soon start to become a reality, namely privacy, profiling (use of facial recognition technology combined with geo-location that
can lead to the creation a digital profile with unwanted personal information), unauthorized augmented advertising and social or ethnical acceptance.

2.1.3 Concepts

A typical AR system framework [Krevelen and Poelma, 2010] consists of tracking, sensing, display and interaction as shown in figure 2.5.

![Typical AR System Framework](image)

Figure 2.5: Typical AR System Framework modules with their respective description.

Each of these modules will now be described in a detailed manner.

**Sensing** By sensing (i.e., capturing) information from the environment, the process of aligning the real and virtual information (known as Registration) takes place and its result is presented to the user. The positioning of virtual objects in the real environment requires an accurate tracking of the users head and detection of other objects located in the environment. In order for this to happen, it is required to have accurate, long-range sensors and trackers and that is the biggest obstacle when building an effective AR system [Azuma, 1997]. In comparison to Virtual Environments, AR has much stricter accuracy requirements and no tracker currently provides high accuracy at long ranges in real time. Specifically, there are three areas, where AR demands more trackers and sensors:

- Greater input variety and bandwidth: due to the inherent requirements of image display, sound generation, among others, VE systems are built mainly to handle bandwidth output. In contrast, its input information like users head/hands location and the outputs of buttons or other control devices is extremely small. AR systems need a greater variety for input sensors and much more input bandwidth [Buxton, 1993] since there are a greater variety of possible input sensors when compared to output displays. Unlike input that can come from anything a sensor can detect, the output is limited to the five human sensors.

Range data is a particular input that is vital for many AR applications [Aliaga, 1997, Breen et al., 1996]. Some AR systems might assume that the entire environment scene is captured at beginning and will remain static thereafter. There are however other applications that require a dynamic environment thus requiring a real-time tracking of real world. In 3D computer graphics a depth map is an image or image channel that contains information of distance of surfaces or scene objects from a viewpoint and the use of that information is enough for some AR systems. However acquiring a depth map in real time is not trivial. This requirement can lead to the use of laser range
finder which is a device that uses laser energy to determine the distance from a device to an object or place. There also some AR applications that require access to a detailed database containing information of the environment. Such information can be considered as a form of input for the system, for example consider an annotation application that displays information in the form of text, images or audio to a tourist (in a smartphone) about every monuments and relevant buildings in Europe.

- **Higher accuracy:** The quality of the AR illusion highly depends on a proper registration of the virtual and real world [Pentenrieder et al., 2006]. One of the main sources of errors for registration is given by the errors in the underlying tracking system and with that being said, registration is only as accurate as the tracker. An AR system requires trackers that are accurate to around a millimeter and a tiny fraction of a degree, across the entire working range of the tracker [Azuma, 1997].

There are several studies comparing various tracker solutions, one of them is from [Pentenrieder et al., 2006] which evaluated four well known optical marker based tracking systems. This study took into account not only the accuracy factor but also another equally important aspect like reliability.

Overall, considering that each tracker has its own advantages/disadvantages, the use of multiple technologies allows to complement each other weaknesses by combining their strengths, and as such, future tracking systems that can meet the stringent requirements of AR will probably be hybrid systems [Azuma, 1997, Council, 1994, Foxlin, 1996, Zikan et al., 1994].

- **Longer range:** trackers built for accuracy at long ranges are scarce due to the fact that most VE applications do not require long range. Two scalable tracking systems for HMD’s have been described in the literature [Ward et al., 1992, Sowizral and Barnes, 1993]. A scalable system is one that can be expanded to cover any desired range by adding modular components to the system. This concept is based on the use of a cellular tracking system which, based on the users location, makes use of nearby sensors to track his position. As the user moves through the world, different sets of sensors will be activated. Even though scalable trackers can be effective, it is a complex system to implement and a potentially expensive one, due to the number of components involved (which vary according to the desired tracking range).

The Global Positioning System (GPS) can be used as part of a long range tracker, however it will not be sufficient by itself (in terms of its accuracy, in a real time mode), regardless of the environment conditions, i.e. terrain, building density and weather. Even when using the Differential Global Positioning System (DGPS) which is an enhancement to GPS, improving its location accuracy from 15-meter to 10 cm (best case scenario\(^1\)), it is still not enough for an outdoor AR tracking system.

Currently, there is no solution for a real time outdoor AR tracking system with the required accuracy and still remains an open problem [Azuma, 1997].

\(^1\)http://www.wikipedia.com (accessed last time on September 22nd, 2014)
Tracking Techniques  The purpose of the Tracking technique is to determine the relative pose (location and orientation) of a camera in real time. This technique allows virtual objects remain aligned with real world objects whenever a user moves his position or viewpoint. It is no surprise then that both tracking and accurate registration between computer-generated objects and real world objects are of the utmost importance when developing an AR application [Rabbi and Ullah, 2013]. [Zhou et al., 2008] categorized the augmented reality tracking techniques in sensor-based, vision-based, and hybrid tracking techniques. The AR Tracking classification can be seen in figure 2.6.

![Figure 2.6: Classification of AR Tracking.](image)

It is also worth noting that tracking is the main issue for outdoor AR systems. [Azuma et al., 1999] described that none of the existing techniques are able to serve as a complete solution this particular tracking variant, leading to the creation of hybrid tracking solutions that combine several technologies in order to fulfill the outdoor AR tracking requirements. Throughout the ten-year development of the International Symposium on Mixed and Augmented Reality\(^2\) (ISMAR), International Symposium on Augmented Reality\(^3\) (ISAR), International Symposium on Mixed Reality\(^4\) (ISMR) and International Workshop on Augmented Reality\(^5\) (IWAR) conferences tracking has been the most popular topic of research [Sowizral and Barnes, 1993]. Even though these conferences are not the only venue for presenting AR research, they are considered the premier conferences for the AR field which provides a good insight of AR research trend and its evolution. The following sections will describe each of the existing tracking methods within the AR domain.

Sensor-based  This technique is based on various types of sensors such as magnetic, acoustic, inertial, optical and/or mechanical strategically placed throughout the environment. None of them is perfect, they all have their own strengths and weaknesses. With that being said, the selection of a sensor comes down to each AR system needs at various levels namely accuracy, calibration, cost, environmental, temperature and pressure, range and resolution. There are numerous studies regarding sensor comparison, for example J. P. Rolland's [Rolland et al., 2001] is one them. Some of the research done in this field has been focusing on creating a hybrid sensor-based version by combining different types of sensors to provide a more robust tracking. [Klinker et al., 2000] developed such sensor by combining body worn local tracking systems with fixed global tracking sensors.

---

\(^2\)http://ismar.vgtc.org (accessed last time on September 22nd, 2014)
\(^3\)http://isar.vgtc.org (accessed last time on September 22nd, 2014)
\(^4\)http://ismr.vgtc.org (accessed last time on September 22nd, 2014)
\(^5\)http://iwar.vgtc.org (accessed last time on September 22nd, 2014)
Vision-based  Vision-based tracking is the most active area of research in AR [Bajura and Neu-
mann, 1995]. In Computer vision methods are used to calculate the camera pose with respect to real
world objects [Azuma et al., 1999]. This task is particularly challenging when facing an unknown envi-
ronment, since it takes some time to collect enough data to be able to deduce the camera pose. In such
environment the system selects the orientation of the coordinate axis at random which may prove to be
inconvenient for the user. In addition, deducing the correct scale solely based on visual observations is
impossible.

One possible solution to overcome these challenges is to add to the environment a predefined sign
and use computer vision techniques to detect and identify it. Such sign is known as a marker and a
computer system can detect it from a video frame and proceed to identify it through a pattern recognition
algorithm. Once the process is complete, it can now define the correct scale and pose of the camera.
This type of approach is called marker-based tracking. Early vision-based tracking used fiducial markers
in prepared environments [Narzt et al., 2006, Shin and Dunston, 2008].

Currently, vision-based tracking research is focused on a markerless approach [Comport and Pres-
sigout, 2006, Chia et al., 2002, Gross, 2003] where any part of the real environment can be used as a
target to be tracked in order to place virtual objects. Both approaches, marker-based and markerless
will now be further described:

• Marker-based: in marker-based tracking, artificial markers, also known as fiducials, are placed in
an environment with each one being a different image (usually a pattern) providing them a unique
property allowing its identification by AR software after being captured by a camera. The pattern
may have various forms, from a barcode to a landscape painting and more recently, a human face.
The evolution of markers is displayed in figure 2.7.

After being recognized, the AR software calculates the correct position, orientation and scale of
a relevant virtual object and embeds it in real time into the real environment on top or near the
marker and as such, if the object is 3D, then by rotating the marker the user can view it from a
360 degrees point of view [Geroimenko, 2012]. Each AR applications can have different marker
detection and tracking requirements. There are three major types of AR markers: digital markers
(for example, an image on a display device), printed markers (for example, a photograph in a
magazine) and natural markers (for example, a human face). AR markers can also be classified
as technical which can be simple geometrical patterns or natural marker like a human face or a
view of the real-world. A detailed discussion involving different markers has been evaluated by
[Zhang et al., 2002].

• Markerless: in order to perform object tracking, markerless augmented reality system relies on
natural features. This kind of approach avoids using and maintaining ambient intrusive markers
which in most cases are not part of the environment. This method also allows for the extraction of
environment characteristics and information for later use.

The most popular and earlier markerless 3D visual tracking system is RAPiD (Real-time Altitude
and Position Determination), described by [Harris, 1993] which served as base for many subse-
Figure 2.7: The evolution of markers in pattern/image recognition technology related to AR applications.

quent vision-based tracking systems. One of the main features of this technique is the fact that it minimizes the amount of data that needs to be extracted from the video feed.

There are several methods developed in this area, each one using a different approach in the camera pose calculation task. For example, in 1998, [Park et al., 1999] presented a method that calculates the camera pose based on a known set of natural features. At the same time it dynamically acquires additional ones and uses them for a continuous update of the pose calculation. This method provides robust tracking even when the original fiducials are no longer in view.

[Vacchetti et al., 2004] developed a second method that combines edge and texture information to get a real-time 3D tracking. Interest points are found in the image for each frame and then matched with interest points of the reference frame which are used for a smooth camera trajectory.

Hybrid Both sensor-based and vision-based tracking have their own limitations and since there is no universal solution for all AR applications, it became necessary to develop hybrid methods. These methods rely on combining both sensor and vision-based tracking methods with the goal of coming up with a solution that minimizes their individual limitations by using multiple measurements to produce robust results. Since hybrid's methods implies the use of multiple tracking methods, the increase of complexity and cost of the tracking system is unavoidable [Rolland et al., 2001].

According to [Azuma et al., 1999] there is not a single technology that is a complete solution for outdoor tracking, therefore a hybrid tracking technique was proposed based on inertial GPS and computer vision technologies. One other solution developed by State et al [State et al., 1996] combines vision-
based tracking (landmark tracking) with sensor based tracking (magnetic tracking). Another example of a hybrid solution is a wide-range tracker based on a component-based framework that combines optical sensor with a vision based approach [Ababsa et al., 2007].

According to [Hughes et al., 2005], a hybrid solution is the most promising way to deal with the challenges of indoor and outdoor mobile AR environments.

**Ideal tracking solution** With so many tracking methods available, a fundamental question arises: which tracking method to use? Unfortunately, as of yet, there is no universal solution for this problem.

**Registration** The purpose of registration is to enable the final alignment of real and virtual information that is presented to the user which can only be achieved through proper tracking. Registration accuracy requires a pixel precision at an interactive frame rate in order to preserve the illusion of both real and virtual world coexisting in the same domain. In many cases, registration accuracy is absolutely crucial in order to achieve the desired results, for example in a biopsy application if the virtual object is not where the real tumor is, the surgeon will miss the tumor, thus failing the whole procedure.

During the registration process numerous sources of errors can emerge. The main sources [Azuma and Um, 1995] are:

- Distortion in the HMD optics;
- Mechanical misalignment in the HMD;
- Errors in the head-tracking system;
- Incorrect viewing parameters (field of view, tracker-to-eye position and orientation, etc.);
- End-to-end delays;

There are two types of error sources: static and dynamic. Of all the error sources mentioned above only the last one, end-to-end delays, is a dynamic error, the remainder ones are static. Despite static errors having a larger number of error sources, the dynamic are the ones that cause most registration errors. In terms of magnitude, studies demonstrated static errors are around 13 mm while dynamic errors can easily exceed 100 mm [Azuma and Um, 1995]. Each of the error sources types will be described next.

**Static registration errors** Static registration errors happen when the user’s viewpoint and the objects in the environment remain completely still [Azuma, 1997], in other words, static registration in an augmented reality image of a real scene indicates how well the system localizes a virtual object in the three-dimensional world when no motion exists in the system [Bryson, 1992]. For example, suppose we want to put an object on top of a table. With imprecise registration, the object might hover on top of the table or even intersect with her. In augmented reality systems that are based on position measurements, one of the primary registration errors comes from position sensor. Magnetic sensors are known
to provide systematic errors (e.g. non-uniform position measurements in many workspaces due to interference from metal objects) and statistical errors (e.g. non-repeatable jitter, i.e. variation of position and orientation caused by noise which yields small displacement of objects which change over time making virtual objects vibrate in the eyes of the user) [Auer, 2000, Holloway, 1995].

**Dynamic registration errors** The largest source of registration errors is the dynamic error caused by end-to-end system delays as the head rotates and translates [Azuma, 1997]. The end-to-end system delay can be described as the time difference between the moment that the tracking system measures the position and orientation of the users head to the moment when the generated images corresponding to that position and orientation appear in the HMD [Azuma and Um, 1995]. This delay is caused the latency inherent of tracking subsystem(s), communication between components, the time it takes for the scene generator to draw the images in the frame buffers and the scanout time from the frame buffer to the displays. The delay can go up to 250ms in slow, heavily loaded networked systems. Being a dynamic error type, this error source is not noticeable when the user stands still, regardless of the gravity of the end-to-end delay. A simple analogy can be done by looking at a photo (being our object in the augmented reality environment), since its a static object, no matter how long we keep our eyes shut (representing the end-to-end delay), when we open them again nothing changed, the image remains at the same position, unaltered. One of the main goals of an Augmented Reality system is maintaining the illusion that the real and virtual worlds coexist. Such task causes a large amount of registration errors and is still nowadays an unsolved problem [Holloway, 1995].

**Display positioning** AR displays can be divided into three categories based on their position between the viewer and the real environment: head-worn, hand-held and spatial. figure 2.8 shows visual display techniques and positioning which will be described in the following sections.

**HMD** Visual displays attached to the head are known as head-mounted display (HMD) or head-mounted projective display (HMPD) [Krevelen and Poelma, 2010]. There type of displays are divided in two categories: optical and video see-through. The optical see-through devices provide an AR overlay through a transparent display while the head-worn capture video from video cameras as a background for the AR overlay, shown on an opaque display, whereas [Azuma et al., 2001]. Cakmakci and Rolland [Cakmakci and Rolland, 2006] give a recent detailed review of head-worn display technology. Current head-worn display technology is still quite limited due to the fact that it is required a direct connection to graphics computer like laptops which have limited battery life. Battery life may be extended by moving most of the computation remotely (for example, using cloud service) and provide wireless connections using standards like IEEE 82.11 or Bluetooth [Krevelen and Poelma, 2010].

**Hand-held** This particular category includes hand-held video see-through displays as well hand-held projectors. They can be described as flat-panel LCD displays that use an attached camera to provide video see-through-based augmentations [Azuma et al., 2001]. They can be in the form of mobile
Figure 2.8: Visual display techniques and positioning.

phones, PDAs or even tablets. This type of devices is minimally intrusive, socially acceptable, readily available, ease of use, with low cost production and highly mobile making them the obvious choice when introducing AR to the masses [Zhou et al., 2008, Azuma et al., 2001]. For example, hand-held video see-through AR acting as a magnifying glass may run on well-established consumer products like mobile phones [Raskar et al., 2001] that show 3D objects, or personal digital assistants with navigation information [Zhou et al., 2008].

Spatial In this approach, the desired virtual information is projected directly on physical objects to be augmented. This thesis focus on this category and as such, this topic will be thoroughly described in next section named Spatial Augmented Reality.

2.2 Spatial Augmented Reality

2.2.1 Introduction

Novel approaches have taken AR beyond traditional body-worn or hand-held displays leading to the creation of a new branch of AR: Spatial Augmented Reality (SAR) providing additional application areas. In SAR, the display system renders the desired virtual information directly onto real world objects. This technology allows for the detachment of the display technology from the user and integrate it into the environment, allowing to solve many of the head or body-attached displays inherent problems such as visual quality (e.g., resolution, lighting, etc.), and human factors (e.g., cumbersomeness, etc.). In its most simple and known form to us all, the augmentation is coplanar with the surface on which they are
projected, i.e. the images could appear in 2D aligned with a flat display surface or even 3D and floating above a planar surface which can be done by resorting to a room-mounted projector with no need for special eyewear. It provides a large field of view and having its graphics displayed at the same distance as the real world objects allows for an easier eye accommodation.

A typical projection surfaces would be a flat or colored wall [Bimber and Raskar, 2011], however there is a more challenging approach which is to have the augmentation displayed on any type of irregular surface. To watch the augmented virtual objects from a viewpoint without distortions, the final image should be pre-distorted based on the geometry of the 3D objects in the real world [Nam et al., 2011]. There are several SAR projects, such as Box\textsuperscript{6} and MIDAS\textsuperscript{7} that clearly demonstrate the potential of this display technology.

As the enhancement of projectors contrast, brightness, dimension and price it is possible for such technique to be further incorporated into even more applications [Wang et al., 2010]. Overall, its capabilities allows for SAR displays to overcome technological and ergonomic limitations of conventional AR systems naturally leading to an increasing interest in exploiting SAR systems in universities, research laboratories, museums, industry and the art community.

2.2.2 Motivation

SAR is a useful technology that can span the gap between the design in virtual space (CAD, 3D images and animations) and the physical space (physical prototypes and 3D printing) by allowing to generate images that are larger than the actual display device virtually anywhere with a potentially higher resolution, brighter images of virtual objects, text of fine details [Jung et al., 2004]. One other unique feature is the fact that this type of display system does not require the user to wear body-attached AR (for example, head-mounted displays) in order to go experience AR. Such features cannot be provided by desktop screens, naturally making SAR a research area of interest. SAR also provides several advantages for the presentation of ideas:

- The physical objects attributes may be altered (color or texture);
- Fine detail can be added;
- User interactions can be simulated;
- Users can physically touch augmented objects;

One of the key components in a SAR system is the use of one or more digital projects often lightweight with low-power consumption. This type of projectors, known as pico projectors use the same technology that powers standard projectors and rear-projection TVs. [Zhou et al., 2008] lists numerous pico projectors with such characteristics while in [D. Zeltzer, 1997] various advantages of spatially immersive displays over head-mounted displays have been noted. The projectors are embedded into the environment and augment physical objects by projecting all sort of information onto them. These objects

\textsuperscript{6}http://www.botndolly.com/box (accessed last time on September 27th, 2014)
\textsuperscript{7}http://www.projectionfreak.com/the-midas-project (accessed last time on September 27th, 2014)
can be tracked, allowing the projections to be updated as the objects move. Such capability allows users to physically touch augmented objects altering their virtual properties while keeping the physical ones static. SAR also naturally accommodates multiple users, supporting collaborative tasks [Thomas et al., 2011].

Several research groups have uncovered the hidden potential of this technology by applying projectors in unconventional ways to develop new and innovative information displays that go beyond simple screen presentations already known to us all [Bimber and Raskar, 2011]. Some of these innovative projects will be described in the following section.

2.2.3 Related work

The SAR concept was first demonstrated by [Raskara et al., 2002] with their work Shader Lamps. The purpose of this work was to augment the existing blank model with computer generated graphics to make the model exhibit realistic lighting, shadow and even animation effects, thus providing it with characteristics previously inexistent. An example of this application can be seen in figure 2.9.

Figure 2.9: The underlying physical model of the Taj Mahal (top left) and the same model enhanced with Shader Lamps (bottom).

Because the approach is to effectively augment the visual properties of the object using a projector, the projectors were called Shader Lamps. Bandy et al. extended it and implemented dynamic Shader Lamps [Bandyopadhyay et al., 2001] which enables users to change the generated graphics interactively by allowing them to digitally paint onto objects with a stylus. Most of the SAR research is inspired by Shader Lamps [Raskar et al., 2001] which has as one of its main goals the prevention of naïve projection of distorted images of virtual objects by introducing compensation based on the configuration of the real 3D object being projected on. There are numerous SAR projects with various applications:

- iLamp [Raskar et al., 2003] supports SAR by casting the pre-distorted image of virtual objects from the projects which can capture as well the 3D environment based on structured patterns;

- [Wilson and Benko, 2010] proposed an interactive SAR system with multiple depth cameras and a projector. The system detects the information of the surrounding environment along with the users motion through depth cameras and display images on the surface objects in a planar table in the
real world for the user to interact with;

- [Linder and Maes, 2010] introduced a prototype robot for a desktop environment which has a robotic arm with a projector and a portable-sized camera to interact with the user;

- [Yang et al., 2001] adopted a concept of an array of multiple projectors to make a huge screen space. Automatically self-configured projectors are able to create huge projection area;

- [Lee et al., 2009] proposed a projection-based information display system for RSAR which displays virtual images in a room with a series of sensors. In [Lee et al., 2009], the robot tracks the viewpoint of the user with the sensors, and then the robot can generate anamorphic (i.e., projection format in which a distorted image is “stretched” by an anamorphic projection lens to recreate the original aspect on the viewing screen), properly distorted images for users on flat-surfaces;

- WARP [Harris, 1993] allows designers to preview materials and finished products by projecting them onto rapid prototype models. The system uses a standard graphical interface, with a keyboard and mouse used for user interface. The projection is not restricted to a fixed area, all feedback is projected onto the tools;

- Surface Drawing [Schkolne et al., 2001] allows a user to sculpt 3D shapes using their hands;

- Spray modelling [Jung et al., 2004] uses a mock-up of a physical airbrush to allow the user to sculpt 3D models by “spraying” matter into a base mesh. The system enhances physical tools by projecting status information onto them allowing to overload a single tool with several functions;

### 2.2.4 Methods

In order to create and maintain the illusion that virtual objects are registered to real objects, several parameters [Raskar et al., 1998c] are required:

- Position of the user;

- Projection parameters of the display devices;

- Shape of the surface of the real objects in the physical environment;

Each of these parameters will be the input for several methods used in the implementation of a projection-based SAR system. The purpose of each of these methods will be briefly described in the following sections.

**Display surface shape extraction** The 3D surface shape extraction can be achieved using a calibrated projector-camera pair [Tsai and Roger, 1986] where structured light patterns are projected and observed by the camera. Raskar et al. [Raskar et al., 1999] describes a near-real time method to capture the 3D shape of the display surface and [Raskar et al., 1998b] describes a unified approach to capture and display on irregular (i.e., non-planar) surfaces [Raskar et al., 1998c].
Rendering and viewing method  Projecting images on irregular surfaces so that they appear correct to a static user have been described thoroughly in [Dorsey et al., 1991, Max, 1991, Jarvis, 1997, Raskar et al., 1998d].

Registration artifacts  In purely virtual environments, where the user is completely immersed inside a synthetic environment without being able to neither view nor interact with the real world, the input requirements of this type of AR system are quite limited, only the approximate position and orientation of the user's head in a fixed world coordinate system are required for a proper AR experience. However, in see-through AR or SAR, preserving the illusion that virtual and real objects coexist requires proper alignment and registration of virtual objects to real objects [Azuma and Bishop, 1994]. Traditional AR methods use body-centric coordinate system to render synthetic objects while SAR methods use a fixed world coordinate system to render them [Raskar et al., 1998c]. In both cases, there are registration errors caused from various sources such as system delay, optical distortion, tracker measurement error all of which are difficult to correct with existing technology [Holloway, 1995]. Since real and virtual objects lie on the same fixed world-coordinate system, static registration errors play an important role in registration from correct estimate of transformations between display devices to tracker and world coordinate system. Actual and estimated perspective projection parameters of the system can result in visible artifacts, in see-through AR these errors result in virtual object "swimming" with respect to real objects [Holloway, 1995] while in SAR these errors lead to different types of visible artifacts [Raskar et al., 1998c].

2.2.5 SAR Limitations

Despite the numerous advantages that this type of display system offers, it still contains significant limitations:

- Depends highly on the display surface properties. A light colored diffuse object with smooth geometry is the ideal. In contrast, it is extremely difficult to render vivid images on highly specular, low reflectance or dark surfaces [Raskar et al., 1998c];

- This type of display is limited to indoor use only due to low brightness and contrast of projected images [Krevelen and Poelma, 2010]. As such, it can only be deployed in controlled lighting environments with restrictions on type of objects with which virtual objects will be registered;

- Other input devices are required for (indirect) interaction;

- Each time the environment or distance to the projection surface changes (inherent to mobile setups) it is necessary to recalibrate the projector. At a first glance this requirement might sound too restrictive, however this type of calibration may be automated by using cameras. This type of calibration can be seen in multi-walled Cave Automatic Virtual Environments (CAVE), where Raskar and colleagues [Raskar et al., 1998a] show how large irregular surfaces can be covered by using...
multiple overlapping projectors, using an automated calibration procedure that takes into account
surface geometry and image overlap [Rekimoto, 1996];

- For front-projector-based SAR, shadows of the user can create problems. The impact of this
limitation can however be minimized by using multiple projectors [Raskar et al., 1998c];

Depending on the AR application requirements, the use of SAR displays might allow to overcome the
technological and ergonomic limitations of conventional AR system. On the other hand this type of
conventional solution has a clear upper hand when, for example, implementing outdoor AR systems.
Thereby, SAR and body-attached AR can be considered as non-competitive systems, i.e. complemen-
tary each other.

2.2.6 The Future

Numerous hybrid solutions have been developed in the area of tracking in AR and they have proven
to be the most robust and complete solution when dealing with the stringent requirements inherent to
some AR systems [Hughes et al., 2005] (outdoor AR environments in particular). The same concept
could be applied in project-based SAR, for example a hybrid environment could be built using digital
light projectors and a see-through HMD. While it would require the user to wear a body-attached AR, this
solution could potentially offer the best of both worlds by combining all of the advantages of conventional
see-through AR with this new SAR paradigm.
Chapter 3

Camera

Vision’s first and foremost requirement is the detection of light from the world. That light can be considered as rays emanating from some natural or artificial source (e.g., a light bulb or the sun), which then travels through space until hitting some object [Bradski and Kaehler, 2008]. When that light strikes the object, most of it is absorbed and the remaining unabsorbed light becomes what we perceive as the color of the light. This reflected light makes its way to our eye (which can be considered as a camera) and is collected on our retina (or imager). The geometry behind this entire process is of the utmost importance to practical computer vision.

In this chapter, a simple but useful camera model called pinhole camera model will be thoroughly described. This model will serve as basis for a more sophisticated model described in section 3.4, a synthetic camera where the intrinsic parameters of a real world camera are taken into account when projecting a 3D world point into a 2D image plane. One crucial concept to accomplish this is the use of homogeneous coordinates, a coordinate system used in projective geometry.

3.1 Introduction

A camera model is a function which maps a 3-dimensional (3D) world onto a 2-dimensional (2D) plane, called the image plane. This function is generally designed to closely model a real-world, physical camera. There are many camera models of varying complexity and one possible way to categorize them is whether or not they are able to capture perspective. The perspective property dictates that objects far away from us appear smaller than objects up close. This is obviously the case with human vision and most cameras in the real world and is a key concept in this thesis.

3.2 Homogeneous Coordinates

Homogeneous coordinates allow the calculation of graphics and geometry in projective space. In other words, it provides the means to project of a 3-dimensional scene onto a two-dimensional image plane. To represent a point in $\mathbb{R}^n$, we use a vector of $n$ elements, encoding what combination of standart
basis vectors is needed to construct the point. With homogeneous coordinates we use a vector of \( n + 1 \) elements instead [Poling, 2012]. This is achieved by adding an additional variable, designated for example as \( w \), into the existing coordinates. Thus, a point defined in Cartesian coordinates as \( P = (X, Y) \) becomes \( P' = (x, y, w) \) in homogeneous coordinates. Furthermore, \( X \) and \( Y \) in cartesian coordinates are re-expressed with \( x \), \( y \) and \( w \) in homogeneous as:

\[
X = \frac{x}{w}, \quad Y = \frac{y}{w}
\]  

(3.1)

One practical example when using these type of coordinates happen in the computer graphics world, where there are, among others, three key operations:

- Scaling;
- Rotation;
- Translation;

For both scaling and rotation operations it is possible to apply the corresponding transformation matrix by simply multiplying it with some vector. For example, for scaling it is defined as:

\[
S(\alpha, \beta) = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}
\]

(3.2)

Applying the transformation scales the vector:

\[
\begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha x \\ \beta y \end{bmatrix}
\]

(3.3)

Applying a rotation to a vector follows a similar approach, however when it comes to translation, it is not possible to implement it with a matrix-vector multiplication. This is where the homogenous coordinates come in and its “magic” is twofold:

- The matrices of existing linear transformations can be easily extended to work with homogenous coordinates. For example:

\[
\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \rightarrow \begin{bmatrix} \alpha & \beta & 0 \\ \gamma & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

(3.4)

- Translation can now be implemented as a matrix-vector multiplication:

\[
\begin{bmatrix} 1 & 0 & i \\ 0 & 1 & j \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + i \\ y + j \\ 1 \end{bmatrix}
\]

(3.5)

All of this is widely used in 2D Computer Graphics and the same principle is also applied a 3D
environment, where a point is represented by a 4-component vector \((x,y,z,w)\), with \(w = 1\) and multiple transformations can be applied by a single 4x4 matrix.


### 3.3 Pinhole camera

A pinhole camera is a simple camera without a lens, where light is projected through small aperture. This aperture appears on the side of a light-proof box as depicted in Fig. 3.1. Light from the scene passes through this single hole and projects an inverted image on the opposite side of the box.\(^1\)

![3D view of a pinhole camera](image)

Figure 3.1: 3D view of a pinhole camera. A solid box containing a small pinhole in the front side. The origin of the coordinate system is set the pinhole, with all 3 imaginary axis represented.

In this simple model, light is envisioned as entering from the scene or a distant object, however, only a single ray enters from any particular point. In a physical pinhole camera, this point is then “projected” onto an imaging surface called projective plane leading to an inverted image, as seen in Fig. 3.2.

![Side-view of a pinhole camera box](image)

Figure 3.2: Side-view of a pinhole camera box: rays of light from the exterior go through the aperture and are projected on the image plane, represented by the back-side of the pinhole box. Due to the aperture being so tiny, only one ray from a single point can go through the aperture. In this particular example, a single ray of light from the top of the house lands on the back-side of the box, designated projective or image plane.

The size of this image relative to the distant object is dictated by a single parameter of the camera: its focal length. This parameter along with few others are represented in Fig. 3.3, where:

- \(f\) is the focal length of the camera, i.e., the distance from the pinhole aperture to the screen;

\(^1\)http://en.wikipedia.org/wiki/Pinhole_camera (accessed last time on September 15th, 2014)
• Z is the distance from the camera to the object;
• X is the length of the object;
• x is the object's image length on the imaging plane;
• p is the principle point, defined as intersection between the optical axis and image plane;

Figure 3.3: A different perspective of the pinhole camera model where two distinct planes are represented: focal plane, corresponding to the front-side of the pinhole box where the aperture is located and the image plane, corresponding to the back-side of the pinhole box where the image is projected.

Looking at Fig. 3.3 it is possible to see by similar triangles that \(- \frac{x}{f} = \frac{X}{Z}\), or

\[-x = f \cdot \frac{X}{Z}, \quad -y = f \cdot \frac{Y}{Z} \tag{3.6}\]

The same principle can be applied when determining the y coordinate. Thus, one can compute the projection of any point on the image plane simply by knowing the focal distance of our pinhole camera and the location of the real world point.

Unfortunately, a real pinhole is far from being the ideal solution to capture images since it does not gather enough light for rapid exposure due to the fact that it only really allows one ray of light from any particular point in the scene which leads to a really dark image [Bradski and Kaehler, 2008]. This is why biological eyes and cameras use lenses to be able to gather more light than what would be available at a single point. The downside, however, of using lenses to gather more light is that it introduces new variables, namely distortions caused by the lens themselves, adding the geometry behind this topic unwanted complexity. Fortunately since in computer graphics there is no need for natural light, one of the existent synthetic camera models used, OpenGL, is based on a pinhole camera which will be later introduced in section 5.3.1.

### 3.4 Synthetic Camera

Despite the fact that the pinhole camera model is fairly simple, it has its own disadvantages, namely the fact that the z value of the house represented in Fig.3.2 is negative and that the projected image ends up being upside down.
The main interest in Computer Graphics (CG) in this topic is the math behind the image projection and it is desireable to keep it as simple as possible, thus, the pinhole camera is replaced by a synthetic camera in which the image plane is placed on the same side of the origin as the scene. Fig.3.4 illustrates the result of this camera replacement, where the main different lies on the fact that projected image now appears rightside up. The point in the pinhole is reinterpreted as the center of projection. Like in the pinhole camera model variant, every ray leaves a point on the distant object and heads for the center of projection, only this time the ray intersects the image plane "sooner" - represented by the point q - since the plane was moved further in the optical axis, in this case \( \hat{w} \). This new image plane is equivalent to the old one, i.e. the resulting projected image of a distant object has the exact same size as the old one and is also at the same distance \( f \). This change alone allows the removal of the negative sign present in equation 3.6 [Bradski and Kaehler, 2008].

At this point it is now possible to determine the basic perspective projection matrix representing a map from 3D to 2D written as a linear mapping between homogeneous coordinates:

\[
\begin{bmatrix}
    x_c \\
    y_c \\
    f
\end{bmatrix} =
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    X_c \\
    Y_c \\
    Z_c \\
    1
\end{bmatrix}
\]

(3.7)

Figure 3.4: A point \( P = (X, Y, Z) \) is projected onto the image plane by the ray passing through the center of projection resulting in a point \( p = (x, y, f) \). In this variant of pinhole camera model, the image plane was moved from the back to the front of the center of projection. While the resulting geometry of point projection behind this change is equivalent, it simplifies the process.

The point at which the optical axis, represented by \( \hat{w} \) in Fig. 3.4, intercepts with the image plane is designated principal point. One would think that the principal point is equivalent to the center of the imager (in the case of biological eyes, the imager can be considered as the retina, in a camera, the image sensor), and while that is true for virtual cameras, for real cameras that would imply that the camera in question was assembled with a micron accuracy but unfortunately that does not happen regardless of the camera nature, biological or mechanical. This is due to a series of factors, from the manufacturing process to the assembling of its components. In order to model the displacement (away from the optic axis) of the center of coordinates in the projection screen, two new parameters are introduced, \( c_x \) and \( c_y \). This leads to a relatively simple model, in which a point \( P(X,Y,Z) \) in the physical world is projected
onto the screen at a pixel location given by \((x_{\text{screen}}, y_{\text{screen}})\), where:

\[
x_{\text{screen}} = f_x \frac{X}{Z} + c_x \quad \text{and} \quad y_{\text{screen}} = f_y \frac{Y}{Z} + c_y
\]  

(3.8)

With this change, there are now two different focal length coordinates. This is due to the fact that the individual pixels on a typical low-cost imager are rectangular rather than square [Bradski and Kaehler, 2008]. Each focal length ends up being the product of the physical focal length of the lens and the size \(s\) of each imager element, where \(s\) is represented in units of pixels per millimeter and \(F\) has units of millimeters, i.e:

\[
f_x = F \cdot s_x \quad \text{and} \quad f_y = F \cdot s_y
\]  

(3.9)

It is important to notice that neither \(s_x\) nor \(s_y\) can be measured directly through any camera calibration process. Same thing applies to the physical focal length \(F\). The only way to obtain these values would be to actually dismantle the camera and measure its components directly. Only the combinations described in equation 3.9 can be derived.

With these parameters, it is now possible to form what its called a camera's intrinsic matrix:

\[
K = \begin{bmatrix}
f_x & s & c_x \\
0 & f_y & c_y \\
0 & 0 & 1
\end{bmatrix}
\]  

(3.10)

This matrix describe the internal parameters of the camera in question. Besides of the focal and principal point coordinates, there is also another variable, represented as \(s\) designated skew coefficient that defines the angle between the x and y pixels. It is customary nowadays to assume that camera's have rectangular pixels so this variable is set to zero during a camera's calibration process. There is also a need to describe the external parameters of a camera, i.e., a way to describe the camera's location in the world and what direction is pointing. One way to represent these two variables can be achieved with a rotation matrix \(R\), and a translation vector \(t\). This forms what its called the camera's extrinsic matrix. This matrix is comprised of a 3x3 rotation matrix in the left-block coupled with a 3x1 translation column-vector in the right.

\[
\begin{bmatrix} R & t \end{bmatrix} = \begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
r_{3,1} & r_{3,2} & r_{3,3} & t_3
\end{bmatrix}
\]  

(3.11)

This matrix describes the transformation of points from world coordinates to camera coordinates. The vector \(t\) can be interpreted as the position of the world origin in camera coordinates while the columns of \(R\) represent the directions of the world-axis in camera coordinates. In other words, the matrix describes how the world is transformed relative to the camera.

With all three coordinate systems involved, camera in equation 3.7, image in equation 3.10 and world in equation 3.11, by concatenating all these three matrices it is now possible to define the projection
matrix from Euclidian 3-space to an image:

\[
\begin{bmatrix}
x_c \\
y_c \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
f_x & s & c_x \\
0 & f_y & c_y \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\
r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\
r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\
\end{bmatrix}
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1 \\
\end{bmatrix}
= C
\begin{bmatrix}
X_c \\
Y_c \\
Z_c \\
1 \\
\end{bmatrix}
\] (3.12)

where $C$ is a 3x4 matrix usually called the complete calibration matrix, with all its values obtainable through a camera calibration process, briefly described in the next chapter.
Chapter 4

Camera-Projector Calibration

As described in 1.2, the camera is one of the components used in this thesis that allows the development of interactive Augmented Reality applications, using its RGB and depth sensors if available, providing important features like skeleton tracking. At a first glimpse, its intrinsic and extrinsic parameters may seem unimportant since the focus of this thesis is on the projection of AR content through a projector. However, in order to prevent the naïve projection of distorted images of virtual objects, it is essential to know its intrinsic and extrinsic properties, so it becomes necessary to go through a camera-projector calibration process.

In the first section, the camera calibration process is briefly described. The purpose of this calibration is not only find camera’s intrinsic parameters but also to find its position in real world coordinates. After this process, it is now possible to calibrate the projector and find its intrinsic parameters and position with respect to the camera. In the last section of this chapter, the results of the whole calibration process are put into use, enabling the estimation of the projector’s position and pose relative to a specific real world coordinate system. The projector positional data will then be used as an input for the SAR application described in the next chapter.

4.1 Camera calibration

As described on the previous chapter, a minimum of 4 unknown variables must be estimated for camera calibration. These variables are the camera model \((f_x, f_y, c_x, c_y)\). Furthermore, the calibration process provides an additional five variables representing the distortion coefficients that describe two types of lens distortion effects: radial and tangential distortion. This raises the number of unknown variables to 9 that represent an unknown transformation in pixel space that can be determined if both start and ending states are known. It is recommended to use an artificial object (with a known size and shape) in the scene in order to make these states easier to identify (although not mandatory, since several approaches for calibrating cameras have been proposed, such as calibration from motion flow). This type of objects are referred to as calibration panels.

The most common type of calibration objects are planar patterns, with features in a 2-D space,
having the third dimension set arbitrarily to some value, usually 0 to facilitate the process calculations.

An example of such planar pattern is the chessboard, displayed in Fig.4.1.

![Chessboard pattern](image)

Figure 4.1: Chessboard pattern with its features already calculated. Each feature is located in the interior intersections where two edges come together and then split off at opposing angles. Also, the origin is set to the bottom left corner but it could just as easily be set to any of other 3 corners.

Given that there are a total of 9 unknown variables that must be estimated for the calibration solution, it would be erroneous to assume that a calibration pattern with at least 9 points would be sufficient. Indeed, [Bradski and Kaehler, 2008] has shown that a planar object by itself is not enough. Rather, a single board can only provide 8 unique equations, therefore at least two boards must be used. However, while it is possible to place two (or more) calibration panels within the image scene, a simpler solution is to use multiple images of the same panel. There are choices to get a different position/pose for the calibration panel: either move the camera around the panel being static or vice versa [Hook, 2008]. In either cases, this process raises an important issue: the world coordinate system is relative to the position and pose of the calibration panel, which differs in each image. It becomes necessary to transform all the calibration panels into a common coordinate system so two vectors are created:

\[
T = \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix}, \quad R = \begin{bmatrix} R_X \\ R_Y \\ R_Z \end{bmatrix}
\]  

(4.1)

\[T\] represents the 3D translation vector while \[R\] the 3D rotation vector, for each calibration pattern in each image. Fig.4.2 depicts this process.

The combination of these two vectors generate the camera’s extrinsic parameters, previously mentioned in 3.11 with one small difference: the rotation vector need to be converted into a 3x3 matrix using the Rodrigues\(^1\) rotation formula, ending up with each image having its own set of extrinsic parameters.

How many unique views of the calibration pattern are therefore needed to solve for all variables? For \(N\) features in each \(K\) calibration pattern, the following inequality must hold true in order to solve for all the unknown variables:

\[^1\text{http://en.wikipedia.org/wiki/Rodrigues'_rotation_formula (accessed last time on October 3rd, 2014)}\]
The left hand side of the first equation represents the number of available constraints across all boards, while the right hand side reflects the extrinsic unique to each board as well as the intrinsics which remain unchanged throughout all the images. [Hook, 2008] states that there can only be 4 unique points worth of data for each board, therefore:

\[ N = 4 \implies K > 1 \]  \hspace{1cm} (4.3)

With these values, two images can theoretically be enough to solve the equation system but that would be extremely susceptible to noise. Taking the opposite approach and using a large number of pattern images would solve this problem, but would also create another one: the computational requirements for iteratively solving a large system of equations can be quite high and there is significant diminishing returns with a high number of images, so overall it also ends up not being worth it. In practice, using 8-12 images is considered to be a reasonable number as it provides a good balance between error reduction and computational requirements. The algorithm used in the camera calibration is based on Zhang’s method[Zhang, 2000] and implemented in Matlab Camera Calibration Toolbox[Bouguet, 2000] in Intel OpenCV library [Zhang, 2000]. The process starts with the detection of \( K \) corners of each chessboard (of known dimensions) per image through the use of a specific algorithm, in this case implemented by [Bouguet, 2000], obtaining a 2D estimation of a total of \( n.K \) points. In each image, the chessboard is set a different perspective. The camera parameters (extrinsic and intrinsic) can then be determined iteratively by minimizing the difference between the real world coordinates and the coordinates resulting from mapping using the camera parameters. The inner-workings of the camera calibration method will not be addressed here as they are not the focus of this thesis and have been extensively studied [Brown,
Determine Camera Position and Pose

Camera calibration is just the first step of a long process that aims at obtaining the projector position and pose (obtained from its extrinsic matrix) with respect to a fiduciary marker placed in a specific position in the real world. This information will then be exported onto Unity, placing its virtual camera as close as possible to the position of the real world projector. The whole (extrinsic) process, described throughout this chapter, is illustrated in Fig. 4.3.

Figure 4.3: Steps taken to find the projector position in relation to a fiduciary marker coordinate system represented by step 5. The point \( P_m \) on the fiduciary marker is seen as point \( p_c \) on the camera image plane, which in turn is seen as point \( p_p \) on the projector “image plane”. The relation between each coordinate system is achieved by applying a rotation matrix and a translation vector to each point.

Once the cameras intrinsic matrix is known from calibration as well the extrinsic matrix of the camera mapping the world referential to the cameras homogeneous referential, it becomes possible to execute steps 1 and 2 from Fig. 4.3. In order to achieve this goal, it is used a modified version of the an application called ArUco\(^2\), a minimal library for Augmented Reality applications based on OpenCV. Among other features, this application can determine a marker’s position and pose relative to the camera (step 1, in Fig. 4.3), represented by a translation vector and a rotation matrix. The use of this application can be seen in Fig. 4.4.

Step 2 will now follow by merging these results into a single matrix and applying its inverse. In other words, this step allows to determine the position of the camera relative to the fiduciary coordinate system. In the next section we will describe steps 3 and 4 in order to achieve the final goal of estimating the projector intrinsic and extrinsic parameters.

\(^2\)http://www.uco.es/investiga/grupos/ava/node/26 (accessed last time on October 4th, 2014)
4.3 Projector calibration

The application chosen to calibrate the projector is an extension\textsuperscript{3} of the Matlab’s camera calibration toolbox previously used in 4.1.

Theoretically, a projector can be seen as a dual of a camera. In practice, there are two main differences that make the calibration of a projector more complicated than that of a camera. The first one is obvious: projectors cannot image the surface that they project upon so that the mapping between the 2D projected points and the 3D projected points cannot be made without the use of a camera. The second one is that it is difficult to retrieve the co-ordinates of the 3D points because the calibrating pattern is projected and not attached to the world coordinate frame in general, unlike what happens in the camera calibration process. This can explain that, to this date, there have been very few practical methods to compute the intrinsic parameters of a data projector.

The goal of this calibration method is to obtain the intrinsic and extrinsic parameters for both camera and projector. Both camera image capture, as well as projector’s image projection, are generally described by a standard pinhole camera model (previously described in 3.3) with intrinsic parameters including focal length, principle point, pixel skew factor and pixel size. Furthermore, calibration also estimates the extrinsic parameters (that include rotation and translation) from a world coordinate system to a camera or projector coordinate system.

The core concept of this method is to consider the projector as an inverse camera. By doing so, it is possible to make use of any standard camera calibration procedure in order to calibrate the projector. The main concern of this method is to find the 3D points of the projected pattern in order to use them together with the 2D points of the image being projected to finally obtain the intrinsics and extrinsics of the projector. The process starts with calibrating the camera, obtaining the intrinsic matrix and a specific extrinsic matrix where the chessboard was placed on a known position. An image containing

\textsuperscript{3}https://code.google.com/p/procamcalib/ (accessed last time on October 4th, 2014)
the chessboard pattern is then projected, generating \( n \) views. The camera then captures the image of this chessboard and [Bouguet, 2000] corner extraction algorithm is applied, generating a set of \( n.k \) 2D points. Using this information the 3D counterparts of these points can now be estimated, using the camera’s intrinsic and extrinsic matrix. It is now possible to establish a correspondence between these \( n.k \) 3D points and the 2D coordinates, obtaining the intrinsic projector matrix as well as all the \( n \) extrinsic matrices generated. [Falcao et al., 2008] offers a more detailed explanation on this subject.

\[
P_c = \begin{bmatrix} R_{pc} & t_{pc} \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\] (4.4)

Step 5 can then be achieved by multiplying these two matrices obtaining matrix \( P_w \):

\[
P_w = C_w * P_c
\] (4.5)

Thus, the resulting matrix describes the projector’s position and pose with respect to the marker coordinate system. This is of utmost importance as it enables scenario \( b \) previously described in Fig.1.1: the marker is originally set in a known position in a wall, so that both projector (as well as Unity’s virtual camera, described in next chapter) and projection surfaces’ position defined in a virtual world are all working within the same coordinate system.
Chapter 5

Unity Platform

The chosen platform to implement this thesis application was Unity. This chapter will introduce the key concepts behind this platform that are essential to understand the inner workings of the application later described in chapter 6.

5.1 Introduction

Unity is a cross-platform game creation system developed by Unity Technologies\(^1\), including a game engine and integrated development environment (IDE). It is a flexible engine that allows the development of video games for various platforms.

Game engines provide all the tools that allow the creation of games. These tools go from the artwork right down to the mathematics that decide what to draw in every frame on the screen.

Unity takes advantage of other software libraries in its functionality, such as Nvidia’s PhysX physics engine, OpenGL, and DirectX for 3D rendering and OpenAL for audio [Goldstone, 2009]. In this thesis, OpenGL will be given a special emphasis since the default *projection matrix* (also called *view matrix*, that defines the frustum of a camera) used by Unity will have to be altered in order to take into account for the intrinsic parameters of the projector previously determined.

5.2 Concepts

There are a series of concepts that are the foundations of any application developed within Unity. Some of these concepts will be described in the following sections.

5.2.1 Assets, Game Objects and components

Assets are the building blocks of all Unity projects. They can represent all sorts of files, from images to 3D models or sounds. Whenever an asset is used, it becomes - following Unity terminology - a

\(^1\)http://unity3d.com (accessed last time on October 3rd, 2014)
GameObject. GameObjects can be considered as containers, while components are the blocks that fill them. All GameObjects contain at least one component known as Transform component that defines the position, rotation and scale of that GameObject in the scene. There are several type of components, they can be used to assign behavior, appearance or influence other aspects of an object's function in the game. This is accomplished by adding components to the target GameObject.

5.2.2 Scripts

Scripts are Unity's key components as they define the behavior of the application. In this thesis application scripts have numerous purposes:

- Apply perspective correction to one more plane meshes by applying a shader to them. Each plane represents a real world projection surface;

- Manipulation of the position of special GameObjects called destination points. These objects represent spheres, positioned around a plane (each contains four points, one per corner) that control both position and pose of the plane. They can either be positioned in the scene by loading their positional information from a XML file or manually by the user's interaction through a drag-and-drop mechanic. This is the very same mechanic that allows the creation of scenario c in Fig.1.1;

- Load/save positional information of each destination control point from/to an XML file;

- Control camera's internal parameters, position and pose by receiving input from an exterior application, containing positional information of the real world projector. This feature would effectively allow the use of a projector mounted on a robot, with the latter supplying its positional information to the developed Unity application;

Each script will be further described in chapter 6.

5.2.3 Coordinate System

The coordinate system used by Unity is based on the Cartesian method, as such, an object $P$ is described in the world scene by a set of three vectors with a $(x, y, z)$ format representing the three properties in the respective axis: translation, rotation and scale values.

5.2.4 World Space vs Local Space

Each object in Unity can be represented using a world or local (or object) space coordinate system. In every 3D world, there is a point of origin, often represented as the position $(0,0,0)$. A world position of an objects is always relative to this point. In contrast, local space assumes that every object has its own referential (and as such, its own point of origin) and, depending on their relationship, other object can be defined relative to this object. Fig.5.1 depicts this situation. This relationship can be in the form
of parent-child composition, meaning one object have one or more objects as its childs with the latter having their referential relative to its parent coordinates referential.

![Diagram of world and local space coordinates](image)

Figure 5.1: Each object in Unity can be represented using two space coordinate system: world and local.

5.2.5 Vectors

Just like points, vectors are described in Cartesian coordinates. They are nothing more than simple lines drawn in a 3D world with a direction and length. Vectors are useful in a game engine context, as they allow to calculate distances, relative angles between objects and their direction.

5.2.6 Meshes, polygons, edges and vertices

3D shape objects are ultimately comprised of interconnected shapes known as polygons. These polygons have a triangle shape and are made up of three connected edges. The location at which these polygons intersect with each other is known as a point or vertex. These locations are used by the game engine to determine collisions between objects. By combining many linked polygons, it is possible to build complex shapes known as meshes.

5.2.7 Material, textures and shaders

Materials set the visual appearance of a 3D model. Materials and shaders complement one another. The first can define the color and textures used in rendering while the latter (defined as a script) is in charge of the style of rendering.

5.3 Camera Model

There is no use in applying perspection correction to an image while using a virtual camera with its own internal parameters that massively differ from the ones used by a real world lens that will be used to project that very same image onto the real world since it will lead to a distorted and out of place image. In order to fix this, it is necessary to apply the results obtained from the camera-projector calibration process (described in chapter 4) namely intrinsic and extrinsic matrices to Unity’s virtual camera.

The extrinsic matrix can be almost directly applied to the Camera GameObject since it already has its own transform component containing the position and pose of the camera. The extrinsic matrix
contains a 3x3 rotation matrix while Unity expects a 3D vector. The conversion can be obtained by using Rodrigues\(^2\) algorithm to the matrix which returns the respective 3D vector.

The intrinsic matrix is not so straightforward since it implies adapting Unity’s projection matrix and make it compatible compatible with the intrinsic matrix previously obtained. As previously mentioned in section 5.1 Unity is based on OpenGL in which one can specify the camera shape (which in turn determines the projection matrix). There are two ways to accomplish this:

- Indirectly specifying the frustum using the method `gluPerspective(fov,aspect,near,far);`
- Directly specifying the frustum using the method `glFrustum(left,right,bottom,top, near,far);`

Each of these two methods will be described in the following sections.

### 5.3.1 Default: `gluPerspective`

Unity by default uses the `gluPerspective` method. This method specifies a viewing frustum into the world coordinate system using a projection matrix based on a series of parameters:

- \(fovY\) specifies the field of view angle, in degrees, in the \(y\) direction;
- \(aspect\) specifies the aspect ratio that determines the field of view in the \(x\) direction. The aspect ratio is the ratio of \(x\) (width) to \(y\) (height). For example, if the aspect value is equal to 2, it means the viewer’s angle of view is twice as wide in \(x\) as it is in \(y\). In the case of the projector used in this thesis work the value would be \(\frac{4}{3}\);
- \(zNear\) specifies the distance from the viewer to the near clipping plane (always positive);
- \(zFar\) specifies the distance from the viewer to the far clipping plane (always positive);

Fig. 5.2 depicts a visualization of the frustum created by this method. The frustum created is symmetrical around the \(Z\) axis. The first argument of this method, designated as \(field\ of\ view\), describes the angle of the apex of the pyramid in the \(YZ\) plane. The second known as the \(aspect\ ratio\), determines the field of view in the \(x\) direction. The near parameter is the distance to the image plane from the origin while the far parameter is the distance to the other side of the frustum, parallel to the image plane. The generated matrix, throughout described in literature\(^3\), is as follows:

\[
\begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & \frac{zFar+zNear}{2zNear-zFar} & \frac{zFar+zNear}{2zNear-zFar} & 0 \\
0 & 0 & -1 & 0 \\
\end{bmatrix}
\]

where \(f = \cotangent\left(\frac{fovY}{2}\right)\) \hspace{1cm} (5.1)

It is important to notice that while the perspective projection matrix is crucial for this thesis work (as it defines the process of creating 2D images from 3D models), this particular OpenGL method is useless

---


Fig. 5.2: On the left: a 2D perspective where “field of view Y” represents the angle formed by the top and bottom of the frustum in the YZ plane. On the right: a 3D visualization of the actual frustum where the aspect ratio is represented by $\frac{\text{width}}{\text{height}}$.

due to the fact that the intrinsic matrix shown in Fig. 3.10, contains five degrees of freedom, $f_x$, $s$, $c_x$, $f_y$, $c_y$ whilst the projection matrix provided by gluPerspective only has two, $\text{fovy}$ and $\text{aspect}$. Fortunately OpenGL offers an alternative to this method - described in the following section - that, after some tweaks, allows the use of the intrinsic parameters of a lens.

5.3.2 Projector-based approach: glFrustum

The goal is to have a projection matrix that can be matched with the intrinsic matrix previously attained in chapter 4.1 and in order to do that it becomes necessary to be able to directly specify the frustum corner coordinates of the virtual camera that is exactly what OpenGL’s $\text{glFrustum}$ method is meant for. It is more difficult and less unintuitive process, however, it provides the much needed flexibility. This method contains six parameters:

- $\text{left}$ specifies the $X$ coordinate of the left edge of the image plane;
- $\text{right}$ specifies the $X$ coordinate of the right edge of the image plane;
- $\text{top}$ specifies the $Y$ coordinate of the top edge of the image plane;
- $\text{bottom}$ specifies the $Y$ coordinate of the bottom edge of the image plane;
- $\text{near}$ and $\text{far}$ are exactly the same as the ones defined in 5.3.1;

The $X$ and $Y$ pairs are typically signed (positive and negative), while the $Z$ pair must both be positive since they measure the distance from the origin. This frustum can be visualized in Fig.5.3.
Figure 5.3: Left and right are measured along the X axis, top and bottom along the Y axis. The blue and green plane correspond to the near and far parameters respectively.

The generated matrix (described by literature\textsuperscript{4,5}) is as follows:

$$
\begin{bmatrix}
\frac{2 \cdot \text{near}}{\text{right} - \text{left}} & 0 & A & 0 \\
0 & \frac{2 \cdot \text{near}}{\text{top} - \text{bottom}} & B & 0 \\
0 & 0 & C & D \\
0 & 0 & -1 & 0
\end{bmatrix}, \quad \text{where} \quad A = \frac{\text{right} + \text{left}}{\text{right} - \text{left}} \quad \text{and} \quad \begin{cases}
\text{left} < 0 < \text{right} \\
\text{bottom} < 0 < \text{top}
\end{cases}
$$

(5.2)

This particular projection matrix is for a generic frustum. If the viewing volume is symmetric, i.e., if \(\text{right} = -\text{left}\) and \(\text{top} = -\text{bottom}\), then it can be simplified to:

$$
\begin{bmatrix}
\frac{\text{near}}{\text{right}} & 0 & 0 & 0 \\
0 & \frac{\text{near}}{\text{top}} & 0 & 0 \\
0 & 0 & C & D \\
0 & 0 & -1 & 0
\end{bmatrix}, \quad \text{where} \quad C = \frac{-\text{far} + \text{near}}{\text{far} - \text{near}} \quad \text{and} \quad \begin{cases}
\text{left} < 0 < \text{right} \\
\text{bottom} < 0 < \text{top}
\end{cases}
$$

(5.3)

From here it is easy to deduce that the only unknown variables are \(\text{right}\) and \(\text{top}\) since \(\text{near}\) and \(\text{far}\) can be ignored, they can be assigned random values as they define the boundaries of the camera’s viewable area.

In a basic pinhole camera it is known [Poling, 2012, Hartley and Zisserman, 2003] that the transformation of a point \([X,Y,Z,1]^T\) in 3D space to a point \([x’,y’,f,1]^T\) is a matrix multiplication followed by a perspective division, i.e., division of the result by the \(f\) component of the result:

$$
\begin{bmatrix}
x’ \\
y’ \\
f \\
1
\end{bmatrix} \approx \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & f & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
$$

(5.4)


\text{5}http://www.songho.ca/opengl/gl_projectionmatrix.html (accessed last time on October 3rd, 2014)
Comparing the transformation that relates the image (pixel) coordinates and the focal plane, described in Fig.5.4, it is possible to derive the following matrix:

\[
\begin{bmatrix}
x \\
y \\
1 \\
1
\end{bmatrix} =
\begin{bmatrix}
k_x & 0 & \frac{c_y}{f} & 0 \\
0 & k_y & \frac{c_x}{f} & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x' \\
y' \\
1 \\
1
\end{bmatrix}.
\] (5.5)

\[
\begin{bmatrix}
x \\
y \\
1 \\
1
\end{bmatrix} \approx
\begin{bmatrix}
f.k_x & 0 & c_x & 0 \\
0 & f.k_y & c_y & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}.
\] (5.6)

It is important to notice that the upper 3x3 portion of the above matrix is in fact the intrinsic matrix (see equation 3.10) obtained when calibrating a camera. In the OpenGL coordinate system the camera is looking down the negative Z-axis so applying the pinhole camera in this coordinate system leads to the same equation as in 5.6, with the exception that in the third column all values are now negative.

During OpenGL's vertex transformation pipeline\(^6\) the perspective matrix is used to transform a 3D point to a normalized device coordinate space (which is obtained by applying perspective division operation), a representation of this space coordinates can be seen in Fig.5.5.

\(^6\)http://www.songho.ca/opengl/gl_transform.html (accessed last time on October 8th, 2014)
Figure 5.5: Normalized device coordinate space, represented by a cube with the front-side and back-side represented by near and far planes respectively. The origin of this coordinate system is set in the cube’s geometric center. Each corner corresponds to the glFrustum method parameters for the near and far planes.

Figure 5.6: Top-view of the frustum, showing the correspondence between the focal and near planes.

Fig.5.6 establishes a correspondence between two key planes: near and focal. The unknown \( s \) is defined as half of the length of the focal plane, and when transformed to the image plane has a value of \( c_x \), so \( s = \frac{c_x}{k_x} \). From Fig.5.6, we have:

\[
right = \frac{s_{\text{near}}}{p} \iff right = \frac{\text{near}.c_x}{p.k_x} \iff right = \frac{\text{near}.c_x}{\alpha}, \quad \text{similarly } \top = \frac{\text{near}.c_y}{\beta}
\]  

(5.7)

Based on these conclusions and replacing the original equation 5.3 (expressed below) a new and final projection matrix can be derived:

\[
\begin{bmatrix}
\text{near} \\
\text{right} \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & \frac{\text{far}+\text{near}}{\text{far}+\text{near}} & -2\frac{\text{far}\cdot\text{near}}{\text{far}+\text{near}} \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\frac{\alpha}{c_x} & 0 & 0 & 0 \\
0 & \frac{\beta}{c_y} & 0 & 0 \\
0 & 0 & -\frac{\text{(far}+\text{near})}{\text{far}+\text{near}} & -2\frac{\text{far}\cdot\text{near}}{\text{far}+\text{near}} \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

(5.8)
It is now possible to apply the intrinsic matrix values obtained in chapter 4 ending up with an OpenGL camera (within Unity) with similar internal parameters as the ones present in the projector used (the calibration process is not perfect as there are some discrepancies that will be later analysed in chapter 7).
Chapter 6

Implementation

6.1 Architecture

This chapter presents a detailed description of the system architecture, as illustrated in Fig.6.1. It is comprised of 6 main categories:

- The **hardware** includes a projector, a Kinect depth camera used for human-robot interactions and a second camera (Logitech 9000 Pro) that serves two purposes: projection calibration and as well as the robots localization system used to detect the presence of fiducial markers in the environment;

- The **Kinect SDK interface**, which allows direct access to Kinects RGB and depth sensors. One of Kinects’ key features employed in this thesis’ work was skeleton tracking;

- **Unity wrapper** enables the use of Kinect SDK internal functions within Unity;

- A **Windows Presentation Foundation (WPF) application** was developed in order to simulate positional information retrieved from a robot localization system;

- **Unity** was the game engine chosen to develop this thesis main application;

- The camera-projector calibration application used was ProCamCalib, developed on **Matlab**, and previously described in chapter 4;

The next section will thoroughly describe the proposed solution that involves all these software and hardware components.
6.2 Homography

Before advancing to the nuts and bolts of the solution implementation, it is essential to describe an essential concept behind this thesis’ work: homography. Homographies are employed in various computer vision and graphics applications [Brubaker et al., 2004]. In the context of this thesis, an homography is used to remove the perspective distortion effect from a perspective (planar) image. When applied to one or more image plane(s) in a virtual scene within Unity, it compensates for the distortion that occurs when the camera is on a non-parallel position in relation to the image projection plane(s). The application on Unity’s camera of the projection matrix (as defined in 6.1), makes it assume the parameters and role of a real world projector. So, in reality it becomes possible to compensate for the distortion caused by a projector with a homogeneous referential not aligned with one or more real world projection surface(s).

The homography concept relies on the homogeneous coordinates: a 2D point \((x_1, x_2)\) in an image
can be represented as a 3D vector \( x = (x_1, x_2, x_3) \) where \( x = \frac{x_1}{x_3} \) and \( x = \frac{x_2}{x_3} \). As stated in 3.2, this is called the homogeneous representation of a point and it lies on the projective plane \( \mathbb{P}^2 \). A homography is an invertible mapping of points and lines on the projective plane \( \mathbb{P}^2 \). Other terms for this transformation include collineation, projectivity, and planar projective transformation [Dubrofsky, 2007]. It is a process where every figure is projected into a projectively equivalent figure, leaving all its projective properties invariant. [Hartley and Zisserman, 2003] provide the specific definition that a homography is an invertible mapping from \( \mathbb{P}^2 \) to itself such that three points lie on the same line if and only if their mapped points are also collinear. They also give an algebraic definition by proving the following theorem:

**Definition 1.** A mapping \( h : \mathbb{P}^2 \to \mathbb{P}^2 \) is a projectivity if any only if there is a non-singular 3x3 matrix \( H \) such that for any point in \( \mathbb{P}^2 \) represented by a vector \( x \) it is true that \( h(x) = Hx \)

In other words, for the estimation of the homography that maps each \( x_i \) to its corresponding \( x'_i \), it is sufficient to calculate the 3x3 homography matrix, \( H \). A representation of such point correspondence can be seen in Fig. 6.2.

![Figure 6.2: Point mapping from one point in a plane onto another point in a different plane. The projection also maps lines to lines as may be seen by considering a plane through the projection centre which intersects with the two planes \( \beta \) and \( \beta' \). Since lines are mapped to lines, central projection is a projectivity and may be represented by a linear mapping of homogeneous coordinates \( x' = Hx \).](image)

It should be noted that \( H \) can be changed by multiplying by an arbitrary non-zero constant without altering the projective transformation. Thus \( H \) is considered a homogeneous matrix and only has 8 degrees of freedom even though it contains 9 elements. This means one needs to estimate 8 unknowns \( H \) matrix parameters. \( H \) is defined as:

\[
\begin{pmatrix}
  x'_1 \\
  x'_2 \\
  x'_3
\end{pmatrix} = \begin{bmatrix}
  h_{1,1} & h_{1,2} & h_{1,3} \\
  h_{2,1} & h_{2,2} & h_{2,3} \\
  h_{3,1} & h_{3,2} & h_{3,3}
\end{bmatrix} \cdot \begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3
\end{pmatrix} \Leftrightarrow \quad x' = Hx 
\]

(6.1)

As previously stated, shape is distorted under perspective imaging: when projecting an image through a projector in a orthogonal position the image appears undistorted, however, when placing the projector in a non aligned position the same image now appears distorted. From Fig. 6.2 it is possible
to reason that a central projection image of a plane is related to the original plane through a projective transformation, and so the image can be considered as a projective distortion of the original one. By computing the inverse transformation and applying it to the image, it becomes possible to undo this projective transformation. The result will be a new synthesized image in which the objects in the plane are shown with their correct geometric shape [Hartley and Zisserman, 2003].

After introducing the mathematics behind homographies, the next sections will address implementation details:

- Description of the scripts developed for the application;
- Description of game objects created for the application;
- Methodology for the correspondence between the set of points \( x, x' \), previously described in equation 6.1, and its implementation in Unity;
- Homography matrix \( H \) estimation;
- Approach for applying the resulting matrix onto a projection surface, defined as a plane in Unity;
- Projection approach for projecting interactive content on each projection surface;

### 6.3 Scripts Overview

The developed application makes use of a collection of scripts, written in C# language, each designed to perform specific tasks:

- **SelectionManager** script initializes all game objects lists, namely source and destination points. Also keeps track of the destination point currently selected by the user;
- **DestinationControl** script updates all projection surfaces homographies matrices whenever the virtual camera moves and/or loads a new set of destination point coordinates from an XML file. Also enables the user to manually control the position of each destination point in the scene;
- **Homography** script calculates and updates the homography transformations applied to each projection surface on the scene;
- **CustomProjMatrix** script alters Unity’s default projection matrix into a custom one. This custom projection matrix was previously calculated in section 5.3.2;
- **ScenarioManager** script allows the user to save or load at any point of the scene a list of destination points, containing one or more projection surfaces;
- **ProxyThreadServer** script allows the update of Unity’s main camera position based on positional information provided by an exterior application. Since the goal is to have Unity virtual camera in similar position as the real world projector the intention here is to be able - into some extent - to simulate a robot’s localization application by sending data through a TCP socket, updating Unity’s virtual camera position in the process;
• **DoOnMainThread** script allows the update of a game object after a network response in Unity. In this case it is used to update the camera game object position, after receiving a new positional message from ProxyThreadServer script;

• **ProjectorProperties** script is used purely for debugging purposes: it simply displays the camera’s current position and pose;

• **InteractionManager** is a *wrapper* used to access Kinect for Windows Software Development Kit internal functions within Unity. It is a free Unity asset\(^1\), called “Kinect with MS-SDK”;

• **PlayerControl** controls the interactive Air Hockey game logic. This game will be later described in this chapter;

• **KinectOverlayer** script updates each paddle position present in the interactive game developed, based on the tracking of specific skeleton joints (for this thesis, one hand of each player as seen on Kinects field of view);

• **CalibrationMode** is a simple script that toggles the visibility of destination control on the scene. Solely employed for testing purposes, since the interest lies in the projection surface itself and not on its destination points;

There is an additional crucial script that represents the *shader* applied to each projection surface (plane game object). A shader is a software program that includes instructions on how the GPU should render certain images on the screen. Shaders have two fairly similar languages: CG (the shading language by Nvidia) and HLSL (the shading language in Direct3D). This topic will be further described in section 6.6.

### 6.4 Game Object Overview

Each virtual projection surface defined is represented in Unity as a parent-child hierarchy of game objects as seen in Fig. 6.3. Each of these virtual surfaces will match a real world projection surface with its coordinates previously measured and stored in the form of an XML file, a process described in section 6.8. An alternative to this method would be, for instance, to detect in real-time a suitable projection surface using Kinects depth map.

Each game object performs a different role, that will now be described:

• **Projection Surface X**, where \(X\) represents the surface number (the application supports up to nine surfaces per scene), it has no components associated, other than the default transform component (previously mentioned in section 5.2.1). Its sole purpose is to store all the game objects related with each other within one projection surface game object parent;

\(^1\)Created by Rumen Filkov: https://www.assetstore.unity3d.com/en/#!/content/7747 (accessed last time October 4th, 2014)
Figure 6.3: Hierarchy representation in the form of parent-child of a Projection Surface game object. Each element represents a unique game object present in Unity scene.

- Plane represents the projection surface on which content will be displayed. It corresponds to a real world projection surface (a wall, for example);
- Source Points stores a set of four X points (see equation 6.1);
- Destination Points stores a set of four X’ points in (see equation 6.1);

Naturally, in order to visualize the scene, it is also required to have a camera. Thus, there is also a game object of type Camera that is treated as the main camera since there is more than one camera present in the scene.

It is also required to have game objects that will represent the AR content to be projected onto each projection surface. Each content to be displayed onto the real world will require its own camera. This topic will be further developed in section 6.6.

### 6.5 Source and Destination Points

The computation of the projective transformation using point-to-point correspondence described in equation 6.1 is now briefly indicated. Local 2D image and world coordinates are selected as shown in Fig.6.2. Let the coordinates of a pair of matching points X and X’ in the world and image plane be (X,Y) and (X’,Y’) respectively. These coordinates are directly from the image and world plane. The projective transformation of 6.1 can be written as:

\[
x' = \frac{x'}{x'3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{6.2a}
\]

\[
y' = \frac{y'}{x'3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{6.2b}
\]

Each point correspondence generates two equations for the elements of the homography matrix H, which after multiplying out are:

\[x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13} \tag{6.3a}\]

\[y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23} \tag{6.3b}\]
These equations are linear in the elements of $H$ [Hartley and Zisserman, 2003]. So it is possible to conclude that four point correspondences are required to fulfill the eight linear equations to solve for $H$ elements. Regarding point restrictions, the sole constraint is that no three points can be collinear. The inverse of the transformation $H$ computed this way is then applied to the whole image to undo the effect of perspective distortion on the selected plane, a process described thereafter in section 6.6.

The four points correspondence takes place in Unity by creating a set of four source points and a set of four destination points. The position of source and destination points game objects determine the values of $x$ and $x'$ used in equation 6.1 to determine the homography matrix $H$. Fig. 6.4 depicts the positioning of these game objects in the scene.

![Figure 6.4: Example of the positioning of both source and destination points in a Unity scene. The position of any destination point can be manipulated at any time. By doing so, a new homography is automatically updated and its result can be instantly visualized. At any point each destination point is "attached" to each corner of the image plane.](image)

The positioning of the destination points can be controlled through either loading an XML file containing their positional data or manually controlled by the user through a drag-and-drop mechanic.

### 6.6 Projection Surfaces

Once gathered both source and destination points’ positions in the virtual world, it is now possible to calculate the respective homography matrix $H$. The estimation process itself is straightforward: a matrix is generated based on the equations 6.3 obtained from four source and four destination points, after which the Gaussian elimination algorithm$^2$ (also known as reduction) is applied to solve this linear equation system.

One could have used however more than four points to estimate matrix $H$. If using $n > 4$ points, the problem of estimating $H$ corresponds to the problem of determining the pseudo-inverse matrix of a $(8 \times 2n)$ matrix. this is equivalent to estimate $H$ from $n$ points in order for minimizing the mean square error.

As previously stated, a shader is applied to each projection surface (built as a plane Game Object that has its own mesh) meaning each vertex composing the mesh of that surface will be affected by any

---

operation defined within that very same shader. With this being said, the resulting matrix is applied to the projection surface plane, through the use of a vertex shader [Lammers, 2003].

Using an internal Unity function, it is possible to transfer the homography values estimated on runtime directly into the shader. This corresponds to the insertion of the desired homography transformation matrix (previously defined in equation 6.1) directly into the vertex transformation pipeline.

The vertex shader pipeline for each projection surface can be seen in Fig. 6.5. OpenGL's vertex transforms is widely described in literature\(^3\), as such, only an overview will be given to describe the pipeline used:

![Vertex Shader Transformation Diagram](image-url)

**Figure 6.5: Vertex shader transformation used for each projection surface. The diagram annotates the transitions between each transform with the coordinate space used for vertex positions as the positions pass from one transform to the next. The resulting homography matrix is computed within a C# script and its value is passed onto directly the shader so that the transformation can be applied to the projection surface mesh.**

- **Object coordinates** is a local coordinate system of objects and is initial position and orientation before any transform is applied;
- **Eye coordinates** are yielded by multiplying Model View Matrix and object coordinates. In other words, objects are transformed from object space to eye space through this method;
- **Clip coordinates** are obtained by multiplying eye coordinates with the Projection Matrix, previously described in section 5.3.2;
- At this point the **Homography Matrix** previously calculated using equations 6.3 is now employed, and multiplied with the previously points obtained within the shader pipeline;
- **Normalized Device Coordinates** (or NDC, represented in Fig.5.5) can be attained by dividing the clip coordinates by \(w\), the homogeneous coordinate. This process is called *perspective division*;
- **Viewport Transform** is yielded by applying the NDC to viewport transformation. The NDC are scaled and translated in order to fit into the rendering screen;

The application was designed so that there can be multiple (up to nine) projection surfaces present in the scene, each with its own homography matrix. This translates into being able to project onto

\(^3\)http://http.developer.nvidia.com/CgTutorial/cg_tutorial_chapter04.html (accessed last on October 5th, 2014)
the real world over nine distinct surfaces at the same time, each with its own position and pose relative to the projector.

6.7 Interactive AR Content

On the scope of this thesis, a small game based on the AirHockey\textsuperscript{4} game was developed, represented in Fig. 6.6. This application integrates augmented reality projected information, with projection distortion compensation, and human gestures recognition to enable interactions between people and undistorted augmented reality content. In a real world, this game has two competing players trying to score points in the opposing player’s goal, using a table having a special low-friction playing surface. In this thesis a virtual table is projected onto the real world where each player controls his own virtual paddle by moving his/her hand up and down (left side player controls his paddle with his/her left hand, same logic goes for the right side player). Each player hand movement is detected by using one of Microsoft Kinect’s features: skeleton tracking [Zeng and Zhang, 2012].

![Figure 6.6: Representation of the Air Hockey game. Each player controls a paddle with the objective of scoring points on the opponent's goal.](image)

Although created in the same scene as the projection surface, it is important to notice that this game is created in a different area and is being viewed by a different camera. This camera then renders what is being seen into a texture that is then applied to one of the projection surface game objects mentioned on last section. This is done so that the transformation referred in the last section can be applied and resulting in an undistorted image when projected onto a real world surface. This process is illustrated in Fig. 6.7, and it is updated at runtime.

\textsuperscript{4}http://en.wikipedia.org/wiki/Air_hockey (accessed last on October 5th, 2014)
Figure 6.7: The game starts using Kinects skeleton tracking feature. One or two players are tracked simultaneously (step 1). The corresponding joint of each player controls the respective paddle in the game (step 2). Camera 2 renders the texture containing the camera view (step 3) which is then applied to one of the projection surfaces and updated in runtime (step 4). The final step can now take place: project onto some real world surface (step 5).

6.8 Saving and Loading States

It is desirable to be able to save and load the list of coordinates containing the position of each destination point which in turn define the position and pose of each projection surface in the world. It was decided to use the Extensible Markup Language\(^5\) (XML), a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The XML file is automatically saved locally or loaded into Unity through the use of specific keybinds. An example of such file can be seen in Fig.6.8.

Figure 6.8: XML file content containing two projection surface coordinates. At any time the user can save or load the coordinates through the use of a keybind (‘S’ key to save, ‘L’ to load). Each time the coordinates are loaded into Unity, every destination point game object position is automatically updated and its corresponding homography matrix calculated.

6.9 Matching Virtual Camera position with Projector’s

Despite the fact that the developed application is not yet fully integrated with the robot’s localization system, the goal has not changed and the intention is to have the application ready to receive input from an exterior application containing the robot’s positional information. As such, a second application was developed in WPF (Windows Presentation Foundation) to simulate the update of Unity’s virtual camera position based on the output of an exterior application sending a message containing position and rotation values. Fig.6.9 contains a screenshot of the application.

The communication process between the application and Unity is done via a network socket, a inter-process communication flow across a computer network. The ProxyThreadServer script previously described (see 6.3) updates Unity’s virtual camera position and pose based on the values contained within the message received.
Chapter 7

Evaluation

This chapter presents the experimental results of the projection of AR content onto the real world. Since it is a process divided in several stages, namely camera and projector calibration ending with the projection itself, each will be evaluated individually over the following sections. In the final section, a brief discussion of the results will take place.

7.1 Hardware Setup

The hardware specification to implement the proposed system is presented in Table 7.1. The proposed system consists of a projector to project the AR content onto real world (planar) surfaces, and two camera, each with its own purpose: the Logitech camera is used for calibration process and also as an localization tool while Kinect provides the much needed depth sensor, essential to develop interactive AR games as the one described in section 6.7.
Table 7.1: Hardware specification of the proposed system.

<table>
<thead>
<tr>
<th>Component</th>
<th>Model</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>Logitech Pro 9000</td>
<td>Video Capture: 640 x 480 at 30 frames/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Focal Length: 3.7 mm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lens Iris: F/2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Image Sensor: 1.9 MP</td>
</tr>
<tr>
<td>Camera</td>
<td>Microsoft Kinect</td>
<td>Sensor: Colour and depth-sensing lenses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Horizontal field of view: 57 degrees</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vertical field of view: 43 degrees</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depth sensor range: 1.2m - 3.5m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data Stream: 640x480 32-bit color at 30 frames/sec</td>
</tr>
<tr>
<td>Projector</td>
<td>Sanyo PLC-XU105</td>
<td>Aspect Ratio: 4:3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Brightness (ANSI Lumens): 4500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolution (Native): 1024 x 768</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Projection Lens: F = 1.7 - 2.5, f = 19.2 - 30.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Projection Distance: 0.94m - 11.48m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Projection Screen Size (Diagonal): 101.6cm - 762cm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optical Zoom: 1.6:1</td>
</tr>
</tbody>
</table>

7.2 Scenario

The experiments conducted were aimed at testing scenario b depicted in Fig. 1.1. Experiments occurred in two variants of this scenario, as seen in Fig. 7.1, where each projection surface was placed in a different angle, represented by $\beta$, with respect to the wall (and consequently, the projector as well).

The goal is to evaluate the system performance when projecting onto small projection surfaces, in this case a simple A4 sheet, at two distinct angles. For each angle, the projection will be tested at three distances from the wall for a total of 11 measurements (4 points each) per angle. AR content will then be projected on the A4 sheet and for each corner of the sheet the offset of the projection will be measured. This process will then be repeated for each position, represented by $P_1...11$ in Fig. 7.1. The entire experimental procedure workflow can be seen in Fig. 7.2.

It is important to notice that despite the fact that the movement of the projector is a manual procedure, information from the modified version of ArUco (see section 4.2) was used. Since the camera is placed on top of the projector, both have synced movement so by measuring the camera displacement, it is possible to determine the projector displacement as well. In a way, one could say that the system works as if the projector was mounted on a mobile platform with an localization system, like a robot.

The update of Unity’s camera position is made by the “Proxy Simulator” application, previously described in section 6.9.

Camera and projector calibration results will be described in the following sections. Afterwards, a
Figure 7.1: On the left, a visualization of the experiment scenario: an A4 sheet is attached to a cardboard, at a $\beta$ angle with respect to the projector. AR content is then projected onto that sheet, measurements then take place between each corner of the page and its corresponding corner of the projected content. This process is then repeated up to 11 times for each $\beta$ angle (the chosen angles were 47° and 60°). The coordinate system used by a fiduciary marker placed in a known position in the real world is used as the systems coordinate system. On the right: the actual experiment taking place.

Figure 7.2: Experimental process workflow.

A thorough analysis of the actual projection results will be given.

7.3 Calibration Process

7.3.1 Camera

The camera calibration procedure (described in Chapter 4) was based on 20 images containing an calibration object, in this case a chessboard, with 9 x 6 features equally separated by 26mm, as already seen in Fig. 4.1.

The auto-focus of the camera had to be disabled as it affect the calibration process by causing erratic readings. Each image filled the photo window frame as much as possible in order to be able to model the entire lens area, otherwise the calibration obtained would be biased meaning it would only be properly calibrated for some area of the lens, specifically the one where the chessboard was placed throughout the calibration process.

The image resolution is 640x480. It was printed on a A4 paper with a 600 DPI laser printer and attached to a cardboard.

Of the 20 images taken, 7 were removed due high reprojection error and the calibration process was restarted. Fig. 7.3 shows the results of the calibration process.
As a way to evaluate the qualitative measure of the accuracy, the calibration process provides a value designated reprojection error that represents the distance between a pattern keypoint detected in a calibration image, and a corresponding world point projected into the same image. The mean reprojection error obtained was 0.156 pixel which is considered to be a fairly good value. The reprojection error for each image taken can be seen in Fig. 7.4, represented as a value and from a pixel point of view.

The camera calibration parameters obtained are the following:
Table 7.2: Camera calibration results.

<table>
<thead>
<tr>
<th>Camera Parameters</th>
<th>Value</th>
<th>Uncertainty (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal length (pixels)</td>
<td>$f_x$</td>
<td>545.04</td>
</tr>
<tr>
<td></td>
<td>$f_y$</td>
<td>547.96</td>
</tr>
<tr>
<td>Principal point (pixels)</td>
<td>$c_x$</td>
<td>344.34</td>
</tr>
<tr>
<td></td>
<td>$c_y$</td>
<td>219.29</td>
</tr>
<tr>
<td>Skew coefficient</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Distortion coefficients</td>
<td>$k_1$</td>
<td>-0.0433</td>
</tr>
<tr>
<td></td>
<td>$k_2$</td>
<td>0.03885</td>
</tr>
<tr>
<td></td>
<td>$p_1$</td>
<td>0.00267</td>
</tr>
<tr>
<td></td>
<td>$p_2$</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

These camera parameters served as the input for the modified ArUco application (see section 4.2) and were the ones that gave the most realistic camera position values and used for the next step: projector calibration.

### 7.3.2 Projector

Just like in the camera calibration process, the evaluation of the results from the projector calibration process are based on the corner reprojection over the images computed. Having used the camera calibration as a preliminary step for the projector calibration, the error of the projection calibration will also depend on the amount of of the camera calibration errors. The error will also depend on experimental setup positioning accuracy and, of course, the corner extraction algorithm. Additionally, just like with camera calibration, special care has to be taken in this step and exclude from the calibration the images that have suffered a bad corner extraction or present high errors for some other reason. The calibration procedure was based on 18 images containing two chessboards, a printed one and a projected one, as seen in Fig. 7.5.

Numerous tests of camera-projection were performed and the experiments were realized in a not-so accurate setup (because of the available material and conditions) but being careful in the corner extraction. One of the issues that initially had a significant impact in the corner extraction was the lighting conditions of not only the room but also of the image being displayed and the camera brightness parameters. Even when using average brightness values on both camera and projector, the images obtained were defective: the corners of the chessboard being projected were not clear and as a result the square corners of the chessboard appeared to be apart from each other, which obviously affected the calibration process in quite a negative way, obtaining reprojection errors four to five times higher than normal.

Besides of the reprojection error, another way to (roughly) validate the results obtained was to visualize the extrinsinc parameters information (see Fig.4.4) and compare it with the position of the camera-projector duo in the real world. The results obtained can be seen in Fig.7.6.

The best calibration using this method presented a mean reprojection error of 1.54 pixel, a value that is considerably higher than the expected: according to [Falcao et al., 2008], this method’s reprojection error is able to go as low as 1 pixel. It is extremely likely that this difference is at least partially due to the
quality of the camera itself. Since the projector calibration process relies on the image quality obtained from each printed/projected chessboard it inevitably affects the corner extraction during the calibration process.

Table 7.3 shows the projector parameters obtained with this calibration method:

<table>
<thead>
<tr>
<th>Projector Parameters</th>
<th>Value</th>
<th>Uncertainty (std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal length (pixels)</td>
<td>$f_x$</td>
<td>1960.91</td>
</tr>
<tr>
<td></td>
<td>$f_y$</td>
<td>2001.19</td>
</tr>
<tr>
<td>Principal point (pixels)</td>
<td>$c_x$</td>
<td>670.88</td>
</tr>
<tr>
<td></td>
<td>$c_y$</td>
<td>696.48</td>
</tr>
<tr>
<td>Skew coefficient</td>
<td>$s$</td>
<td>0</td>
</tr>
<tr>
<td>Distortion coefficients</td>
<td>$k_1$</td>
<td>0.13490</td>
</tr>
<tr>
<td></td>
<td>$k_2$</td>
<td>-0.17003</td>
</tr>
<tr>
<td></td>
<td>$p_1$</td>
<td>0.01846</td>
</tr>
<tr>
<td></td>
<td>$p_2$</td>
<td>0.01648</td>
</tr>
<tr>
<td>Pixel error</td>
<td>$p_e$</td>
<td>1.54831</td>
</tr>
</tbody>
</table>

Table 7.3: Projector calibration results.

In a standard camera lens it is expected to have a near-center principal point. In the case of a projector most often that is not the case. Most projectors have a significant vertical displacement so $c_y$ ends up having a higher value than expected. In this case, with a 1024x768 resolution, the obtained $c_y$ value was 696.48, far higher than it would be if it was near-center point (in which case $c_y$ would be close to 384, half of the respective resolution value). Results also shown that there is also an unexpected significant horizontal displacement, as the expected result would be approximately $\frac{1024}{2} = 512$ and
7.4 AR Projection

After the camera and projector calibration process have taken place it is possible to test the projection accuracy of AR content onto the real world. As previously described in section 7.2, the target projection surface is a A4 sheet attached to a cardboard. The (real world) coordinates of the four corners of the A4 sheet coordinates were set as the destination points in the Unity application (see figure 6.4) and its camera was set to the projector’s intrinsic and extrinsic parameters previously calculated in section 4.3. The content was then projected and the offset of the projection for two different $\beta$ angles was measured and analysed, as described in the following two sections.

7.4.1 Scenario 1

In this section the projection surface depicted in Fig. 7.1 is set at a moderate angle $\beta=47^\circ$. Measurements were taken in 11 positions, 4 points per position so a total of 44 points were measured. The results can be seen in Table 7.4.

It is clear that for all projection points, there is a significant vertical offset. They tend to be in a lower position when compared to the respective target corner, as seen in Fig. 7.7 containing a visual display of the previously shown table. This offset could be caused by any of the two calibration processes or even by mapping the calibration projector model to unity camera model.
Table 7.4: Projection results containing the data for all four corners of the projection surface. Values are measured in cm.

<table>
<thead>
<tr>
<th></th>
<th>Top Right</th>
<th>Top Left</th>
<th>Bottom Left</th>
<th>Bottom Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical point (measured)</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>Projected point (median)</td>
<td>71</td>
<td>73.7</td>
<td>51.1</td>
<td>73.7</td>
</tr>
<tr>
<td>Projected point (std)</td>
<td>0.224</td>
<td>1.149</td>
<td>0.532</td>
<td>1.053</td>
</tr>
<tr>
<td>Error (median)</td>
<td>-0.3</td>
<td>-1</td>
<td>0.5</td>
<td>-1.7</td>
</tr>
</tbody>
</table>

Figure 7.7: Projection results obtained with the projection surface at an angle of 47° with respect to the projector. Each marker x represents the physical coordinate of a corner of the target projection surface (A4 sheet). The remaining markers represent measurements taken at different distances: 1.2m, 1.6m and 2.2m represented by blue star, green cross and red square respectively. The offset values are in cm.

Ideally, the colored markers would be all around the respective target corner (represented by a x), however, that is not the case. They instead appear, for all distances, in an inwards position (relative to each corner) indicating that there is a horizontal scale offset affecting the projection. This offset is most likely erroneous results from the camera and projector calibration since a scale factor also is taken into account when determining the focal values $f_x$ and $f_y$, previously described in equation 3.9 through the scale factors $s_x$ and $s_y$.

It is interesting to notice that, looking at Fig. 7.7, the projection points taken at each distance all stay within close proximity of each other, meaning the projection results do not vary significantly despite the wide distance range (1.2 to 2.2m) used during the measurements.

7.4.2 Scenario 2

In this section, the application is tested to its limits by introducing a steeper angle: 60°. The testing procedure is the same as before. Measurements were taken in 11 positions, 4 points per position so a
total of 44 points were measured. Fig. 7.8 displays the 4 corners of the projection surface targeted and the actual corners projected.

Figure 7.8: Projection results obtained with the projection surface at an angle of 60° with respect to the projector. Each marker represents the physical coordinate of a corner of the target projection surface (A4 sheet). The remaining markers represent measurements taken at different distances: 1.2m, 1.6m and 2.2m represented by blue star, green cross and red square respectively. Values measured in cm.

Unlike previous experiments, the results obtained are far more scattered, meaning the systems calibration is not precise enough to deal with this level of projection difficulty. However, the analysis drawn before from last scenario are also applicable here as it there is also a significant horizontal scale offset present. Table 7.5 presents the statistical data retrieved from Fig. 7.8.

<table>
<thead>
<tr>
<th></th>
<th>Top Right</th>
<th>Top Left</th>
<th>Bottom Left</th>
<th>Bottom Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical point (measured)</td>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>Projected point (median)</td>
<td>74.15</td>
<td>73.7</td>
<td>59.3</td>
<td>73.7</td>
</tr>
<tr>
<td>Projected point (std)</td>
<td>0.608</td>
<td>1.182</td>
<td>0.8934</td>
<td>1.037</td>
</tr>
<tr>
<td>Error (median)</td>
<td>-1.5</td>
<td>-0.8</td>
<td>1.8</td>
<td>-1.4</td>
</tr>
</tbody>
</table>

Table 7.5: Projection results containing the data for all four corners of the projection surface. Values are measured in cm.

7.5 Air Hockey game

As described in the section 6.7, an interactive AR game was developed as a way of showing a possible application of this thesis work.

In Fig. 7.9 the game is seen inside Unity’s virtual scene (left side) and the corresponding camera preview (bottom right). It is a visual representation of Air Hockey’s game workflow described in Fig. 6.7:
the window preview depicts the texture (step 3). The texture is then applied to an projection surface plane object seen by the main camera in the scene, having its output projected onto a real world surface (right side of Fig.7.9). In terms of performance, the application ran at a solid 70 frames per second.

Figure 7.9: The game, the mobile robot, and people playing the air hockey game as projected by the robot on a non-aligned plane of projection.

7.6 Discussion

In the previous sections two calibration mechanisms were evaluated. Starting with the camera calibration, the values obtained were considered quite accurate when testing them with the ArUco application (see section 4.2). The values obtained with this application while using the intrinsic parameters calculated from the calibration method were quite close to the ones achieved by physically measuring the camera’s position.

The calibration of the projector did not prove to be as accurate as the camera one, as the pixel error obtained was quite higher than expected. When taking images for its calibration, it became clear that the quality of the camera had a quite serious impact over the precision of the corners of the projected chessboard. By decreasing overall brightness of both camera and projector it was possible to obtain a higher precision on the corner extraction method in the projector’s calibration process but it was still not enough as the mean reprojection error obtained was significantly higher than expected.

Additionally, two evaluation scenarios were used to test the performance of the projection distortion compensation as well as the quality of the calibration methods used. In the first scenario, where the projection surface was set to a moderate angle (47°), the values obtained were quite promising, specially knowing the projector calibration was far from being perfect. A statistical analysis of the results, together with a visual display comparing the position of the physical corners points with the projected points revealed the existence of a horizontal scale offset, as the projected points appeared in a inward position. This type of offset is likely to have been created when calibrating the camera or the projector, or even both, as the focal values from the intrinsic parameters contain a scale factor within. It also showed that there was a translational offset, also caused by erroneous extrinsinc results from both calibration processes. It was interesting to notice that the distance of the projector to the projection surface had no impact in the quality of the projection.

Finally, in the second scenario, the application was tested to its limits by placing the projection surface
at a steep angle: 60°. It became clear after a visual display of the obtained results that the system calibration is not precise enough to deal with such steep angles. The projection results were far more scattered and less accurate when compared to the ones obtained in the previous scenario. Furthermore, and as expected, the scale offset issue also was present in this scenario.
Chapter 8

Conclusions

This thesis focused on exploring the topic of Spatial Augmented Reality by implementing a SAR system specifically designed to be deployed on mobile platforms. The goal is to close the gap between two distinct fields: Robotic and Augmented Reality.

The system developed allows the creation of unique interactive AR scenarios that otherwise would not be possible. Virtual planar surfaces are created within a virtual environment through which they are projected onto real world surfaces through a projector. An algorithm is then applied to each virtual surface so that the perspective distortion caused by its position and orientation with respect to the projector is compensated, leading to a distortless image. The system is scalable, allowing simultaneous projection over nine distinct real world projection surfaces, each with its own position and orientation.

The platform on which the system was developed was adapted so that the intrinsic parameters of the projector used for the projection can be applied to a virtual camera present in that very same platform. In order to achieve the desired results, the system required a two-step calibration envolving a camera and projector. Starting with camera, its intrinsic parameters allow to determine the properties of its lens. These values served two purposes: find the camera position with respect to a fiduciary marker placed in a known real world position which would serve as the coordinates referential of the developed system, as well as an input for the projection calibration process, in which the projector is takes the inverse role of a camera (since it projects images instead of capturing). Besides determining the projector intrinsic parameters, the projector calibration process also provided crucial extrinsic parameters establishing a relation between the camera and projector positions. Through this relation and the camera-marker extrinsic relation it was possible to determine the projector position with respect to that very same marker, concluding the entire calibration process.

Even though the system was not initially deployed on a robot per se, the system was adapted so that it could receive positional information from exterior systems, which could be, for example, a robot’s localization system. For compensating - to some extent - the lack of a robot, a second application was developed in which positional information of the projector can be sent directly to the system, updating the virtual camera position in the virtual world, leading to a recalculation of the necessary compensation to all the projection surfaces. In a later stage of this thesis, it was possible to adapt a robot with a mounted
projector to test this application, in particular the Air Hockey interactive game.

In order to evaluate the performance of the developed system, two scenarios were evaluated where a projection surface was placed with two distinct angles, 47° and 60°, with respect to the projector. The target surface was placed at a known position in the real world and AR content was then projected upon it from 11 different positions, in which a total of 44 projected points were measured. An analysis of the results was made for each scenario: regarding the first scenario, the results revealed the presence of a horizontal scale error caused by the camera-projector calibration process, however, the results obtained were considered to be quite promising. In the second scenario, with the projection surface placed at a steep 60° angle, the results obtained proved to be less accurate and it became clear that the quality of the calibration obtained was not precise enough to deal with the projection onto projection surfaces at such steep angles.

8.1 Limitations

The initial idea in this thesis was to not only compensate for the projection of AR content over real world oblique surfaces but also to be able to have the projector in any position and pose. Unfortunately, due to lack of time, the latter was not implemented as the system requires a reference plane (for example a wall, as seen in scenario b in Fig.1.1), to which the projector needs to be in a perpendicular position in order to properly compensate for the distortion caused by projecting onto oblique surfaces. This is a crucial feature for when the developed system is to be deployed on a robot, as such, it represents a serious limitation of this thesis work that we, however, expect to solve in future work.

8.2 Future Work

This work provided the foundations to deploy a SAR system on a mobile platform, however, further work is required in order to be able to deploy it on a robot:

- Adapt the developed system so it can interface with a robot localization system;

- Determine the point from which a robot rotates and use that information to rotate the system’s virtual camera. If, however, the camera used for projector calibration is the same used for inferring the robot localization, then this step is not required as our work allows the immediate determination of projector position and orientation from that of the camera. This would allow to have a robot to project from any pose;

Furthermore, it would also be important to find a method to minimize the calibration errors from both camera and projector devices to further improve the precision of the projection.
Bibliography


P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: a class of displays on the
reality-virtuality continuum. In SPIE Proceedings: Telemanipulator and Telepresence Technologies,

C. Nam, M. Sung, J. Lee, and J. Kim. Plane-dominant object reconstruction for robotic spatial augmented
reality. In 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pages
627–630, November 2011.


Proceedings of the International Workshop on Augmented Reality : Placing Artificial Objects in Real

K. Pentenrieder, P. Meier, G. Klinker, and M. GmbH. In Dritter Workshop Virtuelle und Erweiterte Reali-


I. Rabbi and S. Ullah. A Survey on Augmented Reality Challenges and Tracking. Acta Graphica, 24:

R. Raskar, H. Fuchs, G. Welch, A. Lake, and M. Cutts. 3d talking heads: Image based modeling
at interactive rate using structured light projection. Technical report, University of North Carolina at
Chapel Hill, 1998a.

approach to image-based modeling and spatially immersive displays. In Proceedings of the 25th
Annual Conference on Computer Graphics and Interactive Techniques, pages 179–188. ACM, July
1998b.

Reality (IWAR'98), pages 11–20, 1998c.

R. Raskar, G. Welch, and H. Fuchs. Seamless projection overlaps using image warping and intensity
blending. In Proceedings of the 4th International Conference on Virtual Systems and Multimedia,
November 1998d.

R. Raskar, M. Brown, R. Y. W. Chen, G. Welch, H. Towlse, B. Seales, H. Fuchs, and G. Herman. Multi-
projector displays using camera-based registration. In Proceedings of Visualization, pages 161–522,


