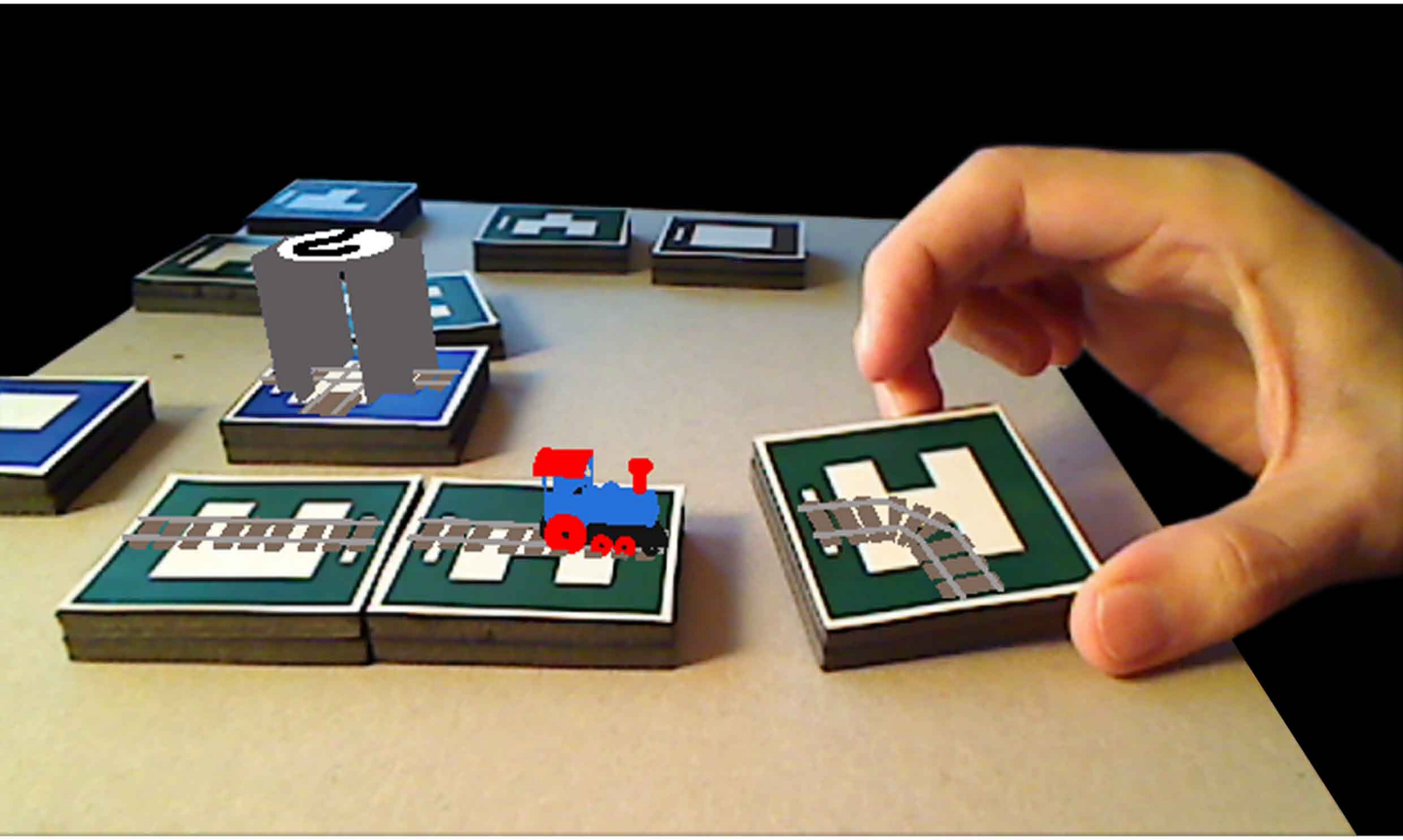

Computer Aided Board gaming using Computer vision

Lasse Farnung Laursen
Supervisor: Jon Sporring
Department of computer science
Copenhagen University

Contents

Preface

This thesis was developed as the final masters project at the Department of Computer Science, University of Copenhagen. I have received help, support and useful feedback from a number of people during its course of development, whom I would like to take the opportunity, to thank.

First of all, I would like to thank my supervisor Jon Sporring, for his continuous feedback. I also owe a significant of my gratitude to Daniel Wagner, for his support and expertise in the field of pose estimation. Furthermore I would also like to thank Jonas Christian Drewsen, Simon Brettschneider for their feedback regarding technical aspects, and testing techniques. I also wish to thank my brother, Thomas Frøhlich and Bue Vendel-Larsen for discussing and suggesting overall improvements. My brother deserves additional thanks for taking his time, to create nice front page layout for the thesis.

Additionally I would like to thank all my willing test participants, who were kind enough to take time out from their busy schedule, and help me test my prototype.

My mom and dad also deserve my gratitude for me getting this far and finally, my sister deserves a world of thanks for putting up with my presence, helping me pilot test my prototype, providing me with the Eye of Judgment, and giving me a roof over my head, during the last months of my thesis.

Thank you!

Abstract

Augmented reality allows for an almost seamless fusion of the real and virtual world. The myriad of possibilities this provides, for both classic board games, and modern computer games, is inspiring. By using augmented reality, the user can interface with the game by physically interacting with real objects. Providing the player with this kind of tactile interface essentially integrates the "controller", as a part of the game.

In this thesis I implement and evaluate a system for the real-time tracking of fiducial tags, along with homography based pose estimation. The thesis provides an introduction to projective geometry theory, and uses it as a basis to describe the calculations and techniques necessary, for the real-time tracking and pose estimation of multiple tags. The systems robustness is tested, and the pose estimation algorithm is compared to an open source computer vision library. An informal comparison is also conducted with a commercial game which uses augmented reality. The advantages and disadvantages of augmented reality are considered when using it to enhance both classic board games and modern computer games. To further study the challenges and possibilities of using augmented reality in games, four game concepts are considered, and one is implemented as a rough prototype. The prototype is tested with ten willing participants to reveal problems and interaction issues, by using the "think aloud" protocol.

The thesis highlights one approach to pose estimation, which is successfully integrated into a working game prototype. The subsequent user tests shed light on how the interaction with the system is perceived by the user, and what potential problems exist when using augmented reality to create a tactile interface for games.

1 Introduction

Computer games and board games exist in different worlds. One world is digital and the other analogue. Both of these mediums have their individual strengths and weaknesses. Computer games are capable of presenting the player with an almost life-like reality, which can immerse her deep into a fantasy world, where anything can happen. Board games, not capable of presenting such an immersing world, are often limited to using symbols and static elements, leaving it up to the players imagination, to shape and animate it. However, no matter the medium in which the game exists, the goal is the same. To provide each player with an enjoyable experience.

I believe that both board games and computer games stand to gain from each others unique features, and provide an overall improved experience. A board game stands to gain from the computers accountability, computational power and feedback devices (screens and speakers). A computer game could benefit from a board games more direct interaction with tangible game elements, and overall social experience. The most vague of the aforementioned benefits is the social experience, which warrants further clarification. As computer games have evolved, they have also grown more sociable. Today, computer games exist that allow for more people to play together, than previously thought possible. But the experience of a multiplayer computer game, and a multiplayer board game, is not the same. A board game puts players close together and often right in front of each other. Interactions are performed in between players and each participant is within view of the others. On the other hand, interactions in computer games are performed through the computer, and players are rarely within view of each other. These two types of games provide an all together different social experience. By selecting positive aspects of both types, I believe the users experience can be improved.

Augmented Reality is a field of research concerned with the combination of real-world and computer-generated data. Using it, along with other computer vision techniques, I intend to combine elements from both classic board games and modern computer games. To properly combine both real and virtual elements, I will implement a system capable of tracking fiducial tags, and individually estimating their poses, in real-time. The system is to be implemented in a prototype game to gain insight as to how well the interaction possibilities, provided by augmented reality, work. The robustness of the developed tracking and pose estimation will be tested, and the developed prototype game will be played by a number of volunteer players. Results from both tests will be discussed and evaluated. In addition to testing the software implemented during the course of the thesis, an informal test and comparison is made with an existing commercial product, which uses augmented reality.

I hope that this combination of classic and modern game elements, forming a kind of hybrid game, will provide the player with a unique experience and enjoyable experience. Through the development of the thesis, I also hope to gain valuable insight into the implementation and use of augmented reality, and how this can shape human computer interaction. As for whether or not augmented reality will have a significant impact on the gaming industry is hard to predict, but I believe that we are only now seeing the tip of the iceberg.

1.1 Thesis Aim

The main focus of this thesis is to present and implement a system capable of tracking fiducial markers, and estimating their individual poses in real time. The intended use for

the developed system is an augmented reality game. Therefore, it has also been a goal to consider what possibilities the use of such a system has to offer in a gaming context. To properly test the developed system, and further study the possibilities, a prototype game has been developed which makes use of augmented reality.

A list describing the individual specific goals is provided below:

- Research and implement computer vision techniques relevant for this thesis.
- Identify and describe the computer vision techniques used in the project.
- Design and construct a system to track and interpret board game elements in real-time.
- Evaluate the robustness of the developed hybrid game system.
- Consider possible enhancements to the implemented techniques based on the robustness of the developed system.
- Evaluate and analyze the users experience and interaction with the hybrid game.

1.2 Implementation and Design Limitations

This section details implementation and design limitations in this thesis:

- **Prototype Game** - The game developed in the master thesis serves as a proof of concept and is intended as a means to test interaction and computer vision techniques. It is not intended to become a fully developed commercial title, but rather a working prototype.
- **Closed Environment** - The prototype will only function within specific predetermined parameters.
- **Hardware Specific** - The prototype is only guaranteed to execute on the hardware it was developed. Especially in regards to the cameras it can use, as each have their own set of internal parameters.
- **Computer Vision Robustness** - Computer vision is an enormous field of research and it is beyond the scope of this master thesis to handle each conceivable issue, when dealing with a camera. Areas of complications include camera calibration, image distortion and/or lack of proper lighting. Although these issues cannot be ignored, they are of low priority, in comparison to the the main focus of this thesis.
- **Computer Vision Aids** - Providing the computer with additional visual aids (apart from the board game it is tracking,) would likely improve tracking stability and perhaps even simplify the algorithms computing the pose of the different game elements. However, additional aids would put further constraints upon what the computer must have in its field of view, and in a worst case scenario hinder player interaction with the game. Consequently, the proposed system will by design not make use of any additional visual cues, apart from the markers visible within the playing area. A benefit of this approach, is that the hybrid game would not require any additional components apart from the game itself, its pieces, a PC, and a camera.

- **Computer Vision Complexity** - The implemented system will only make use of a single camera and display all visualizations on a standard PC screen. The camera is not necessarily static, but its purpose is to view the entire game area at all times. Mounting the camera on the user and providing the user with a head mounted display is an ideal expansion, but beyond the scope of the thesis. It is considered as future work in Section 11.4.

- **Pose Estimation Algorithm** - It is beyond the scope of this thesis to analyze and compare several existing pose estimation algorithms. A single pose estimation algorithm, based on homographies and the extraction of the cameras location and orientation from it, has been researched and implemented in this thesis. A description of the algorithm and why it was chosen as opposed to other algorithms is described in Section 6.

1.3 Thesis Disposition

Apart from the two immediate sections following this introduction, the thesis is composed of three large segments. The first segment (Sections 4 through 8) contains as few implementation details as possible as to allow the described system to be implemented on any platform providing similar capabilities. The second segment (Sections 9 and 10) explains implementation specific details along with any theory related specifically to the implementation. The third and final segment discusses results obtained from the testing procedure and sums up the accomplishments of this thesis in a final conclusion (Sections 11 and 12).

Although the thesis contains a number of sections which hopefully everyone should be able to understand, the intended audience is a student, with at least a bachelor in computer science, and a general understanding of computer vision. The thesis is intended to be read sequentially. It is possible to read only select chapters or skip complete sections altogether, but references back to previously explained theory only appear to clarify and avoid ambiguity. With that in mind, here is a brief explanation of sections and their contents:

Section 2 defines common terms and the mathematical notation used throughout the thesis. Section 3 details a number of projects related to this thesis. Section 4 more thoroughly introduces the notion of augmented reality in relation to this project, and how it can be used to improve the gaming experience. Section 5 provides the reader with the basic theory regarding projective geometry, and Section 6 explains how the pose estimation algorithm works based on the previous theory. In Section 7 a number of possible game ideas are explored which make use of augmented reality. The actual concept chosen and implemented as a prototype is also explained including its rules, user-, and computer controlled elements. Section 8 provides the remaining details needed in order to properly combine the real and virtual reality. The most significant implementation related details are explained in Section 9. An overview of the implementation is provided in Section 10 with a UML diagram and a brief description of all the classes in the developed code. Finally, Section 11 describes the testing procedure used to evaluate the prototype developed during the thesis. The chapter also discusses the results and details ideal improvements to the developed prototype. Section 12 ends the thesis by summing up the most notable accomplishments.

2 Definition of terms

For the convenience of both myself and the reader, I have chosen to include this section in the thesis. If the reader is well versed within the topic specific notation, this section can be skipped.

In general the terms and mathematical notations in papers and books regarding projective geometry and related topics conform to the same standard. There are however instances where some authors have decided to exchange one commonly used letter describing a matrix, for another. In most cases I have chosen to stick with the notation I have found in other texts. Only where I have discovered ambiguity or believed to improve the notation, have I deviated from set standards.

The list below comprise the defined terms for this thesis, and the following subsection elaborates on the mathematical notation used.

- **Computer Game** - Often also referred to as a video game, referring to the raster display device often used in classic digital games. I use the term computer game which is intended to include games running either a computer or on a console (such as an X-Box, Super Nintendo Entertainment System, Neo-Geo, etc.).
- **Augmented Reality (AR)** - A concept involving the combination of real-world and computer-generated data. The real-world data (often an image of the real-world) is augmented by adding (or even subtracting) data to (or from) it. A common usage is that of a heads-up display (HUD) being overlayed on a surface, close to a person. The HUD could relay such information as position and heading, or additional data regarding elements shown in the image.
- **Virtual Reality (VR)** - A computer-simulated environment that allows for user interaction. The environment often only provides a visual and audial feedback, which can be accomplished by having the user look into screens, and listening to speakers. Modern VR systems also allow for the user to experience tactile sensations.
- **Pose Estimation** - Determining the pose of an object in one (or several) images, relative to some coordinate system.
- **Radio Frequency IDentification (RFID)** - An identification method making use of devices such as RFID-tags and transponders. RFID-tags exist both as active and passive variants. The active version contains its own power supply, where as the passive version relies on a transponder signal, in order to function and relay its identification.
- **Tangible User Interface (TUI)** - A user interface, in which a user interacts with digital information through a physical environment.
- **Heads Up Display (HUD)** - Any transparent display which presents data, and can be monitored concurrently with a primary task. An example is the information shown to pilots in military grade airplanes.
- **Head-Mounted Display (HMD)** - A head (or helmet) mounted display device. Placed on the users head, the device has a field capable of visualizing graphics in front of one (monocular HMD) or both eyes (binocular HMD).

- **Tag** - A surface with an easily recognizable pattern upon it. It is sometimes also referred to as a fiducial marker. A good example of a recognizable pattern is the surface of a domino brick, which has white dots on it.
- **Ideal Points** - Points considered to lie at infinity in projective geometry. Often expressed in homogeneous coordinates (see Section 2.1).
- **Homography** - A homography is a one-to-one relation between two sets of points. The relation can be expressed as a matrix, and can be used to the define the relation between points, in two images.
- **Degrees of Freedom (DOF)** - The number of independent variables required to specify the state of an element. For example, a point in a two dimensional space ($\mathbb{R}^2$) has two degrees of freedom (x and y), which must be specified, before its exact state (position), is determined.

2.1 Mathematical Notation

The following notes should be observed regarding the mathematical notation in this thesis:

- **Vector/point orientation** - All vectors are assumed to be in the upright vertical notation unless explicitly written horizontally. For example, the vector $\mathbf{v}$ is therefore equal to $(x, y, z)^\mathsf{T}$ and $\mathbf{v}^\mathsf{T} = (x, y, z)$.
- **Euclidean and homogeneous notation** - The letter E, affixed as subscript to a point, will define it to be expressed in Euclidean coordinates. Thus, a point $\mathbf{p}$ expressed in Euclidean coordinates will be written as $\mathbf{p}_\mathrm{E}$. If not specified, the point is assumed to be expressed in homogeneous coordinates. To avoid superfluous notation, only points and vectors adhere to this notation. Matrices and other mathematical operations are assumed to function in the same coordinate system, as the points and vectors they affect, unless otherwise specified.

3 Previous/Related Work

As previously stated, the aim of this thesis is, in part, to develop a system, which can combine elements from both the real and a virtual world. It is also my intention to perform preliminary research and analyze the users interaction with a game, which uses this technology. I have chosen to divide the work I have researched for this thesis into two groups. First, projects and papers with a focus on technique and research regarding pose estimation and augmented reality. Second, projects and papers concerning games and/or gaming which uses augmented reality or similar technology.

There are undoubtedly more projects related to this thesis than the ones listed in these two sections, but the ones listed here are those that I found to be most relevant.

3.1 Computer Vision and Augmented Reality

The projects and papers listed in this section form the basis of my research involved in this thesis. They are listed in the order, in which I discovered them. The list is not exhaustive, but is limited to the projects which I studied most and/or are most related to this master thesis.

A number of these projects served as initial inspiration for attempts at creating a working pose estimation algorithm. However, none of the projects or papers below provided a description of the process from start to finish, which I understood. Therefore, it has become a personal goal of mine, to describe the pose estimation process, in as much detail as possible, in this thesis. Although the projects below did not result in a directly applicable algorithm, they have been a great help in the form of inspiration, especially in regard to the shape and look of the tag.

- **Matrix: A Realtime Object Identification and Registration Method for Augmented Reality [Rek98] by Jun Rekimoto** - This paper introduces a technique for producing augmented reality systems that simultaneously identify real world objects (via tags) and estimate their coordinate systems. The method presented in the paper utilizes a 2D matrix marker (tag), and a square shaped barcode, which is uniquely identifiable and aids the system in estimating the coordinate system. The method presented in the paper is the precursory technology to Cybercode [RA00] developed by Jun Rekimoto and Yuji Ayatsuka. Cybercode has been used in recent commercial title named "The Eye of Judgment" [Wikd] for the PlayStation 3. This article and the related commercial product was the initial inspiration for this master thesis. Initial versions of the tag used in this thesis is based upon the tags shown in this paper.

- **ARToolKit [Kat] by Hirokazu Kato et al.** - The ARToolKit is a software library for building Augmented Reality applications. It provides the developer with practically everything necessary to integrate augmented reality into an OpenGL Application. A now discontinued project exists which has successfully integrated the library with Ogre3D (the graphics engine used in this thesis). The ARToolKit uses an iterative closest point (ICP) algorithm in which a type of brute force approach is applied, varying parameters and evaluating whether an improved result was achieved or not. It is written in the `C` programming language and made available freely for non-commercial use under the GNU General Public License. Because the ARToolKit is an open-source project, I spent a while investigating the code performing the actual pose estimation. Unfortunately, at

the time of writing, it is virtually without comments, and the provided documentation is almost completely lacking, in regards to the details behind the actual pose estimation. Additionally, the code is written in a very minimalistic style (many variables are denoted by a single letter) which makes it even more difficult to understand. As a result, this project has had little effect upon the master thesis although it has been researched extensively.

- **Studierstube Tracker [WAM$^+$] by Daniel Wagner et al.** - Studierstube Tracker is a computer vision library for detection and pose estimation of 2D fiducial markers. It is the successor to a project known as ARToolKitPlus based partially on the ARToolKit and some aspects of ARTag. The Studierstube Tracker uses a non-linear pose estimation algorithm (with Gauss-Newton iteration) and is closed source. Although neither the studierstube tracker nor the ARToolKitPlus has had a large impact on this thesis, apart from further tag design inspiration, the project is mentioned due to the contact I have maintained with Daniel Wagner during the development of the thesis. He graciously provided help and insight into a number of issues plaguing my pose estimation algorithm.

- **Planes, Homographies and Augmented Reality [Lil03] by Björn Liljequist** - Björn Liljequist's master thesis presents a system for estimating camera movements in image sequences where the scene is known to contain significant planar structures. Apart from initial manual interaction, the system is fully automated and can compensate for variable camera focal length as well as new planes becoming dominant in the scene. This master thesis is similar to my own, in that it also uses a homography based approach to pose estimation. However, there are a number of crucial differences between this, and my own thesis. Some of the differences include: Liljequists project is not dependent on real-time execution, there is no running user interaction, the tracking is not done via tags, and there is only one set of points for which a pose is estimated. Although an explicitly mentioned goal of the thesis was to

  > [...] achieve and convey a complete understanding of the process.

  , there were a few intricate details that I did not discover in the thesis. Possibly due to some of the fundamental differences between the uses of the pose estimation. Regardless, the project has been a useful aid in understanding a large portion of the theory relating to projective geometry and its uses.

- **ARTag [Fia, Fia04a, Fia04b] by Mark Fiala** - Similar to the ARToolKit, ARTag is also a software library designed to be used in new or existing applications adding augmented reality functionality. ARTag is closed source and at the time of writing, development is currently halted due to licensing issues with the National Research Council of Canada. This project served as yet another source of inspiration regarding the tag design, as it conveniently provides a comparison of most tag designs in use. The tag detection used in this project is more robust that then one utilized in the ARToolKit [Kat] according to the paper. It is an edge based approach as opposed to pattern recognition, and does not rely on gray scale thresholding. Use of this algorithm is considered as a future improvement in Section 11.4.

3.2 Augmented reality Gaming

According to Bimber and Raskar, Augmented reality was "born" in 1968 [BR05] when Sutherland introduced the worlds first head mounted three dimensional display [Sut68]. Although augmented reality has been applied within a number of industries, gaming has not been one of them until recently. In order to gain inspiration and a better understanding of the possibilities I have researched some of the earliest and latest developments regarding augmented reality in games. The list is sorted according to the date of appearance.

- **1000CS Game System [Kni] by Virtuality** - The first wide spread virtual reality game system introduced in the nineties by a company called "W Industries" (later renamed to "Virtuality"). The game system was mainly targeted for amusement locations, and not as a commercial home product for the end user. It is mentioned here because of its historical significance in relation to augmented reality. In contrast to augmented reality, virtual reality aims to substitute the whole of reality with a virtual one. The visual and audial senses of the user receives stimulus almost exclusively from a simulated environment, giving the illusion of another world. Both virtual and augmented reality share similar roots in technology, and I have been unable to discover an augmented reality related game product prior to the 1000CS game system. It is also the first virtual reality game which I had the opportunity to enjoy.

- **Virtual Boy by Nintendo** - The only commercial virtual reality gaming system launched by a high-profile company intended for home-use. The console received lackluster reviews and quickly faded from the interest of the gaming industry without gaining any wide spread acceptance [Ken01]. Just like the 1000CS game system, it is only historically related to augmented reality and mentioned because I was unable to find a commercially released augmented reality console. At the time of writing I have only been able to find a now discontinued commercial product [Ltd].

- **Collaborative gaming in augmented reality [SEG98] by Szalavári et al.** - Szalavári et al. introduce a local collaborative augmented reality environment for home based entertainment. The system uses HMDs to visualize a Chinese board game (mahjongg [Wikf]) playable by up to a total of four players. In contrast to the prototype developed in this thesis, players interact with the virtual game via "Personal Interaction Panels" [Sza99] which resembles a notebook-sized hand held panel and a pen. The project is an excellent example of how a board game can be ported to a digital medium, with almost no modification to the original game concept.

- **ARQuake: an outdoor/indoor augmented reality first person application [CDS+00] by Ben Close et al.** - One of the first games utilizing augmented reality as a core concept connecting the player with the physical world she is in. Based on Quake [Wikl], a first person shooter for the PC platform, the developed application expanded on the game by adding augmented reality. The real and virtual world are synchronized giving the illusion that both the player and her opponents exist in the same world. The project uses multiple technologies to allow for outdoor/indoor augmented reality, among others a modified version of the ARToolKit [Kat].

- **Human Pacman: A Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction over a Wide Outdoor Area [CWG+03] by**

Cheok et al. - A game where the player moves actively around in both the real and virtual world. Based on Pac-Man [Wiki], the game allows for multiple players to participate. One player assumes the part of pac-man and must eat all the dots in the "level". The other players assume the roles of the ghosts and must catch pac-man.

- **Heuristics for Tabletop Games [Köf07] by Christina A. Köffel** - This diploma project presents a comprehensive set of heuristics for the evaluation of tabletop games. According to the author, the heuristics

  > [...] contain all facets offered by tabletop games, such as game play and game story, virtual interface and the special conditions of augmented tabletop games.

  The heuristics described in this project have been taken into consideration during the development of the prototype game in this thesis.

- **Eye of Judgment [Wikd] by Sony** - A commercial game by Sony released in 2007 utilizing augmented reality as part of the gameplay. It has received mixed reviews [Met] and sold approximately 200,000 copies not including sales in Europe [vgc]. Although Sony has a number of titles available for its consoles utilizing the camera, I have chosen only to include this title in the list as it is their only title using augmented reality to an advanced degree. This product was a part of what originally inspired me to select augmented reality as a topic for my master thesis. An informal comparison between the developed prototype and the Eye of Judgment is conducted in Section 11.3.

- **Levelhead [Oli] by Julian Oliver** - A spatial memory game created by Julian Oliver. Among other libraries the game utilizes the ARToolKit [Kat] for pose estimation. The object of the game is to lead the main character out of a series of rooms represented by cubes in the real world.

  One of the few existing augmented reality games which, in my opinion, makes good use of the technology to provide a unique experience to the player. The game has served as additional inspiration for this thesis.

4 Augmented Reality and hybrid gaming

This section provides a brief introduction to augmented reality and why combining classic board games and computer games along with augmented reality is an ideal concept.

The different projects mentioned in the previous section makes it apparent that augmented reality is not a new phenomenon. Neither is virtual reality, with which it shares a number of technological elements. At the time of writing, previously launched pure virtual reality products seem to have all but disappeared from the industry. The lack of continuing success, of the CS1000 [Kni], is undoubtedly tied to the struggling arcade industry [Ken01]. But even the commercial home product Virtual Boy has not had a successor or comparable product since its initial release.

Due to a number of reasons, I believe augmented reality has a greater chance of success of becoming an integral part of the gaming industry. The average consumer has far more computational power at her disposal when compared to the early nineties, making a product for home use within the realms of possibility. Augmented reality aims to enhance and not substitute the existing reality which allows an application to take advantage of a users pre-existing knowledge regarding that reality. Players, being indigenous to the real world, have certain preexisting concepts regarding it. This could be utilized by the application to allow for a smoother and transition free understanding of various game elements. Finally, because augmented reality is based on the actual world, components from the real world can be used as a basis for interaction. This concept is described in further detail in the itemized list below. The list contains, what I believe to be, some of the most significant advantages of combining board games and computer games, and using augmented reality in the process. Because the advantages relate differently to either type of game (board or computer based), I intend to clarify how it relates to each of them individually.

- **Haptic Feedback** - As mentioned above, objects in the real world can be used as a basis for interacting the game. As long as the object can be recognized by the computer, it can be merged with the virtual world. Because the object still exists in the real world, the player is free to touch and affect it. The user can be given a tangible user interface where an effect in the virtual world may have its cause in the real one. I believe that this removes a barrier otherwise created by controllers or other interfaces, where the device being manipulated rarely resembles what it affects in the game. An example of this, is the standard game controller used when playing racing games. The controller looks nothing like the steering wheel normally used to control the car. However, a number of controllers and technologies exist, apart from augmented reality, that reduce this barrier between the real world and the game world. The most common one is the rumble feature found in most gamepads. It affects the player via vibrations, usually intended to signal specific events. Another less common example, are controllers that more or less accurately resemble the object manipulated in the game. For example, the plastic guitar used in Guitar Hero or steering wheel controllers used for racing games. The most advanced example is perhaps a force feedback steering wheel which, in addition to resembling an actual steering wheel, provides the player with realistic haptic feedback. Via the controller the player is able to both determine how much traction the in-game car as well as what type of surface it is driving on. However, even though these controllers may resemble or feel like their "actual" counterparts, I believe the user is well aware that they are not one and the same. Using augmented reality can

make an object the user physically manipulates an integral part of the game.

A good example of haptic feedback, using augmented reality, is the game Levelhead [Oli]. Players manipulate a character in a room by turning and rotating a solid cube they hold in their hand. The cube appears to contain a room within which the character resides. If the player tips the cube in one direction, the room will also tip, and the character will move in that same direction.

From the perspective of a computer game, augmented reality allows for an object to become an actual part of the game, thereby providing haptic feedback. From a board game perspective augmented reality will allow for an object to move from symbolizing an element, closer to being that element.

- **Accountability** - Briefly mentioned in Section 1, the computer helps provide accountability for actions taken within the game world. In general board games tend not to include elements that can be easily disputed among players, as there is rarely an objective third party involved. A computer could be utilized either as the final judge in case of disputes, or just as kind of consultant providing its "opinion" on the situation. Using the computer for the purposes of accountability is not limited to areas of dispute. The computer can assume the burden of tracking tedious game elements, such as score or other statistics. In a similar fashion, the computer can also be used to prohibit cheating. This advantage is only relevant in relation to board games since accountability is the very foundation for most computer games.

- **Hidden Information** - Just as the computer can take care of tracking tedious game elements, it can also handle information intended to be kept hidden from one or multiple players. Board games and computer games each have their individual strengths and weaknesses regarding how to handle hidden information. In the case of board games, hidden information is usually limited to simple actions, or information in the form of cards not visible to the player(s). After all, the players must know the rules in order to play the board game, so any actions the board game requires to be performed, must be undertaken by its players. A computer could assist in controlling elements of the game, which the player should not have intimate knowledge of, there by hiding information crucial to the games progression.

  In computer games, the player is usually presented with information on a useful-to-know basis. It is common to hide a large amount of information from the player, since it either has little relevance for her, or ruins the gaming experience if revealed. There are however occasions where information should be readily available for one player, but not for another. This is often the case in multiplayer games where players are up against each other. If the game being played uses multiple display devices, the solution is straightforward. Only reveal relevant information on the screen, used by the intended recipient. Unless the game is being played over a network, games usually only make use of a single screen. In this case, how to display this private information is not as straightforward. In case of a single screen, most modern computer games avoid this type of partially hidden information altogether. Some games leave it up to the players to handle who views the information presented and when. Using augmented reality, board game elements such as individual cards can be seamlessly integrated into the gameplay sidestepping the issue altogether. The Eye Of Judgment [Wikd] is a good example of how this can work in practice. Each player hold a number of cards hidden from the

others, while still playing with the same computer. If only one player is participating in the game, she can play against the computer. It has a hidden deck that the player cannot see, and plays by a set of rules that the player is not immediately aware of.

- **Participation** - Using augmented reality allows for a potentially seamless integration of a computer opponent into a board game. Computer controlled elements can exist exclusively in the virtual world and still inform players of their status, without burdening them with the task of manipulating them. If speakers are available, the computer can communicate its intentions aurally as well as visually. In addition to creating an artificial intelligence which can interact with other players, the computer could also act as a surrogate player, allowing for a person in a distant location to participate in a local board game. The Eye Of Judgment [Wikd] allows for all types of play combinations: Player Vs. Computer, Local Player Vs. Player, Remote Player Vs. Player, and Computer Vs. Computer. Most computer games already allow for computer opponents to participate, and for players to join a game from a remote location.

- **Assistance** - Board games require a set of rules to be learned before actual play can begin. Computer games also adhere to a number of rules. The key difference being that with board games, the players themselves must enforce, and therefore know the rules. But even though computer games only inadvertently allow the players to perform unauthorized actions, most computer games have a set of mechanics, which the players must learn in order to fully enjoy the game. It is common for modern computer games to contain a tutorial explaining most of the key concepts, there by engaging the player in play as soon as possible. This also relieves the player of reading a lengthy explanation of how the game works, and possibly turning the player away from the game before any play has begun. Using augmented reality, this concept can be incorporated into board games. The computer can help the player complete a tutorial and/or continuously provide support during game play. I believe this could help improve the learning curve for almost any given board game. If individual displays are used, the assistance can be individually tailored and hidden from other players. Szalavári et al. [SEG98] experienced that the help provided in their augmented reality game was used extensively by the players.

- **Increased Social Experience** - Board games are almost by definition a social experience, since few can be played alone. On the other hand, single player computer games are quite common, as are multi-player computer games. I believe that by shifting a portion of the game out into the real world, computer games can be made more sociable. Both for player, as well as non-players.

  In between players the social interaction is increased because the players are potentially as close to each other, as with a board game. Individual elements of the game can be interacted with by either player, possibly even at the same time. Players can read each others reactions and interact much closer than in a computer game. For example, if two players are engaged in an online strategy game, they can often communicate vocally. But using only vocalization still leaves room for interpretation and can lead to misunderstandings. If both players are situated in-front of each other, each player can literally show the other what they intend.

  For by-standers (non-players), the situation is slightly different. In computer games, the player is often in control of what information she wishes to see. The player continuously

interacts with the game to find the information which is most important at any given time. Bystanders who are not actively participating will therefore have a hard time grasping what exactly is going on, especially if they have no previous knowledge of the game being played. If some of the game elements are present in the real world, the player can still keep her focus by only looking at what interests her, and bystanders are free to examine which elements they find interesting. If the surrounding people are more aware of how the games is played and how the player is doing, I believe this will encourage more interaction between the player and non-players.

It is important to note that these advantages are not necessarily exclusive to the combination of board games and computer games using augmented reality. They are ideal advantages that can otherwise be hard to achieve depending on the medium (analogue or digital). For example, in the fictitious online strategy game briefly mentioned before, each player could be given a set of in-game tools to improve coordination in between players. One such tool could allow each player to draw directly on the virtual battlefield to convey their intention. Improving the interaction in between players would make the game more sociable, without combining it with board game elements or using augmented reality.

Although the use of augmented reality can provide a number of advantages based on the situation it is applied in, there are also some possible disadvantages that should be considered.

- **Reliability** - Depending on the software and applied technology, the augmented reality provided by the computer may be flawed or unreliable. This thesis, and a number of the projects related to it [Oli, Kat, Fia, RA00, WAM$^+$], use a recognizable pattern to perform pose estimation. Consequently, if the pattern is not recognized, the computer is unable to perform the pose estimation. This can either be caused by objects obscuring the pattern or improper lighting. Objects obscuring the pattern can be compensated for by using multiple markers [KB99] or multiple image sources. Improper lighting is tolerable if a robust detection algorithm is used. The impact of both these pitfalls can be reduced, but neither of them can be completely avoided. If no marker is visible, the computer cannot perform pose estimation, and if there is no light, the computer will be unable to see anything.

- **User comfort** - Depending on how the augmented reality is presented to the user, the experience can lead to feelings of discomfort. A number of AR projects require the user to wear head mounted displays, which could potentially lead to eye strain depending on the screen visualizing the virtual environment. Moderate use and high quality equipment would reduce this problem. Köffel et al. mention user comfort as one of the evaluation heuristics in their project [Köf07].

- **Reduced social experience** - Although augmented reality can be used to make games more sociable, it should also be noted that the opposite effect can also occur. Board games which previously required multiple human players to play, could potentially now be played alone. In addition, augmented reality applications utilizing head mounted displays may exclude other by-standers from participating, or even properly observing the events unfolding making the experience in general less sociable. Providing an abundance of head mounted displays, or an alternate public display of the augmented reality, would minimize this issue.

The above list only notes some of the potential disadvantages regarding augmented reality. The problems discovered during the testing procedure for the game prototype, developed for this thesis, is detailed in Section 11.

5 Projective geometry

This section gives an introduction to projective geometry and other related concepts relevant for the understanding of this thesis. Readers with a thorough knowledge of projective geometry and camera representation may skip this section.

The examples provided in the section are assumed to be in a two-dimensional space unless otherwise stated. Two dimensional examples are much easier to visualize than three dimensional ones. The three dimensional case is often just a generalization of the two dimensional one.

5.1 Euclidean and Projective Geometry

Euclidean geometry, attributed to the Greek mathematician Euclid, is the geometry that most people are familiar with. It helps describe the shape of objects, lengths and angles in-between lines and/or other shapes. Projective geometry can be considered similar to euclidean geometry, except that it is much less restrictive. Formally, euclidean geometry is considered to be contained within projective geometry. However, euclidean geometry is not what we see when we view the world. We see a perspective projection of the world. A projective space within which almost everything changes depending on the perspective it is viewed from. For example, from most perspectives, a circle looks like an ellipse. Depending on the perspective, a square can look like a trapezoid. One of the few things that remain constant during a perspective projection is the the concept of straightness. Straight lines remain straight, no matter where they are viewed from.

In euclidean geometry, parallel lines never meet by definition. Most parallel lines we, as humans, perceive in the world seem to meet in the distance. A good example of this are train tracks. Two lines that undeniably run side-by-side and (ignoring complex train crossings) never meet. Yet when viewed from most angles, the train tracks always seem to meet at the horizon. This is one of the key differences between euclidean and projective geometry. In euclidean geometry, two lines almost always meet in a point. The only exception to this are lines that are parallel. But exceptions can lead to complications, and in many cases it is easier to simply state that all pairs of lines intersect at some point. Lines that are parallel are claimed to intersect at infinity. According to Hartley and Zisserman "(...) this is not altogether convincing, and conflicts with another dictum, that infinity does not exist, and is only a convenient fiction" [HZ04]. A solution to this problem is to extend the euclidean space with these points at infinity and referring to them as **ideal points**.

By introducing these points at infinity, euclidean space is transformed into projective space. A space which in many aspects is similar to the familiar euclidean space, yet also includes these new ideal points at which parallel lines meet.

5.2 Homogeneous Coordinates

In classic euclidean geometry, a point is represented by a pair of coordinates (x, y). Euclidean space is homogeneous and there is no distinct origin. An origin can be chosen at will and the relation between two or more points simply depends on the chosen coordinate frame. As established, euclidean and projective geometries share a number of similarities. But points placed at infinity is a concept restricted to projective geometry. It is useful to distinguish which points are at infinity and which are not. Homogeneous coordinates allows us to do

exactly that.

A point in euclidean geometry (x, y) is represented as $(x, y, 1)$ in homogeneous coordinates. Any multiplication by a non-zero scalar k yields a new set of coordinates which, is an equivalent representation of the same point:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = k * \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} k * x \\ k * y \\ k * w \end{bmatrix}$$

Therefore, in order to arrive at the euclidean representation of the point, all that is required is to divide every coordinate with w, yielding the euclidean representation:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \\ w/w \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$

A question which springs to mind is what to do when $w = 0$. In ordinary arithmetic (using only real numbers), dividing by zero has no meaning. This is where homogeneous coordinates are especially useful in regards to projective geometry. Points where $w = 0$ are considered to be ideal points (and lie at infinity). Since dividing something by zero yields a meaningless expression, this prohibits ideal points, mathematically, from entering pure euclidean space which fits perfectly with the theory. Extending this concept into a third dimension is just a matter of adding an additional coordinate. In the two dimensional case just described, all the ideal points can be considered to form a line at infinity. In the three dimensional case consisting of 4 coordinates (x, y, z, w), the line at infinity becomes a plane.

5.3 Representing Cameras

Recognizing and tracking elements in the real world is an integral part of this thesis, which means cameras and related theory is as well. The following text provides an introduction to existing camera models.

If three dimensional space is to be approximated by a two dimensional representation, a projection is needed. A camera is usually the medium chosen to conceptualize how a projection is performed, both mathematically and practically. Not only are cameras a good tool in understanding how a projection is performed, but they are also essential in the equivalent real world process.

It is important to distinguish between the two types of cameras related to this thesis. Real-world CCD (Charge coupled device) cameras and virtual cameras. The theory and mathematics described in this section relate to both types. For the CCD Cameras, the theory is a sufficient approximation of how cameras operate in the real world. For virtual cameras, it is the actual theory applied in modern rendering applications and graphics hardware, when simulating a camera.

The cameras described in this section are known as **pinhole cameras**, also shown in Figure 1(a). They are an ideal representation of a camera, with an infinitely small aperture size. Therefore, every ray of light (or projection) entering the camera and projecting an image to the back of it must travel the same one point in space, known as the camera center. In Figure 1(a), the camera center is where the two red lines meet. Cameras in the real world do not have an infinitely small sized aperture, but use optical lenses in order to focus the light to a single point. These cameras are therefore referred to as finite cameras. The pinhole camera

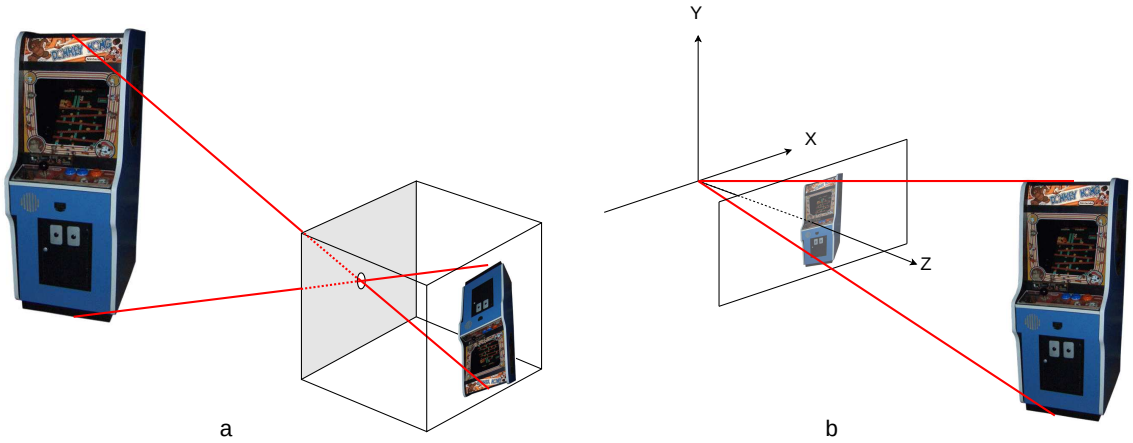

Figure 1: **Camera Models.** (a) A pinhole camera with an infinitely small aperture. (b) A model equivalent to the pinhole camera. Image of Arcade Machine credited to Driggs et al. [DB].

model does not describe geometric distortions or blurring of unfocused objects, as can occur with a finite camera. But it is an acceptable approximation within the bounds of this thesis. As a further simplification, the viewplane (on to which the image is projected) will be placed in front of the camera center, as shown in Figure 1(b). This makes the model much easier to describe, and hopefully understand.

Figure 2 shows a more detailed version of the pinhole camera model. In the figure, it can be seen how the point $\mathbf{i}_{\mathrm{E}}$, which exists in a three dimensional euclidean space ($\mathbb{R}^3$), is projected on to a two dimensional euclidean space ($\mathbb{R}^2$). For simplicity sake, the camera center is used as the origin of the coordinate system. The point $\mathbf{i}_{\mathrm{E}} = (i_x, i_y, i_z)^{\mathsf{T}}_{\mathrm{E}}$ is mapped to the image plane as follows:

$$(i_x, i_y, i_z)^{\mathsf{T}}_{\mathrm{E}} \rightarrow (fi_x/i_z, fi_y/i_z)^{\mathsf{T}}_{\mathrm{E}} \tag{1}$$

As mentioned earlier, homogeneous coordinates solves the problem of perceiving points at infinity, yet still allowing points to exist both in euclidean space and projective space. Readers familiar with graphics rendering will know that homogeneous coordinates are especially useful since they allow for many types of transformations to be expressed in matrix form. This allows for a number of various transformations (translations, rotations, skew and projections) to be combined into a single matrix. Homogeneous coordinates also allow the operation performed in eq. (1) to be performed using a matrix:

$$\begin{pmatrix} i_x \\ i_y \\ i_z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fi_x \\ fi_y \\ i_z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} i_x \\ i_y \\ i_z \\ 1 \end{pmatrix} \tag{2}$$

As eq. (2) shows, the resulting calculation using the matrix is essentially the same as in eq. (1). The homogeneous coordinates ensure that the first two coordinates are divided by last, in order to return the point to euclidean space. This is shown in the following equation:

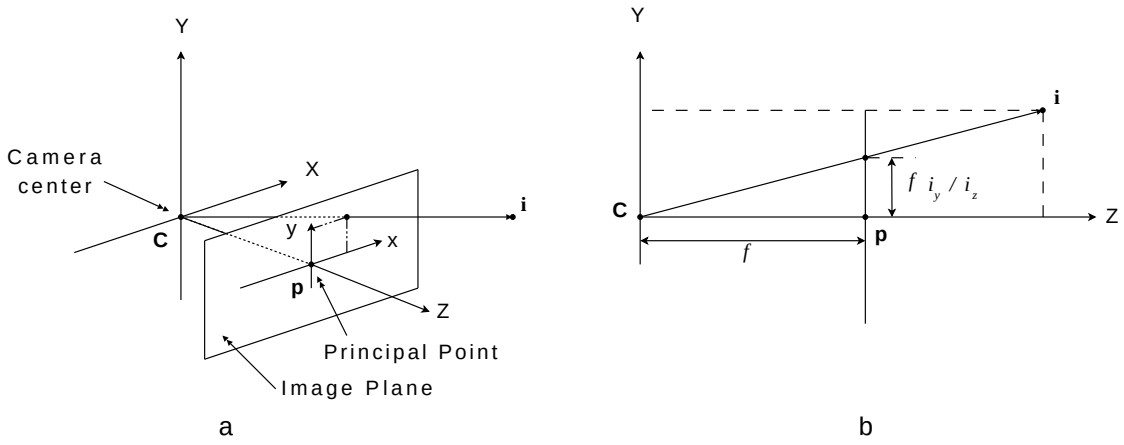

Figure 2: **Pinhole camera geometry.** (a) A detailed isometric perspective of the pinhole camera model shown in figure 1(b). **C** is the camera center and **p** is the principal point. The point being projected onto the image plane is **i**. (b) Is a side view of the same model as presented in (a). As the figure shows, the y coordinate of the point **i** on the image plane is calculated by the equation $f * (i_y/i_z)$. f being the focal length of the camera, and i_y/i_z is the ratio of inclination of i_y over i_z from the camera center.

$$\begin{pmatrix} fi_x \\ fi_y \\ i_z \end{pmatrix} \rightarrow \begin{pmatrix} fi_x/i_z \\ fi_y/i_z \end{pmatrix}_{\mathrm{E}}$$

The 3x4 matrix in eq. (2) is a very simple homogeneous camera projection matrix containing the cameras intrinsic parameters. A more accurate and complex version of this model is detailed in the following subsection. In addition to the matrix containing the intrinsic parameters, it is also necessary to define where the camera is located. How this is accomplished is explained in Section 5.3.2.

5.3.1 Intrinsic Parameters

The intrinsic parameters of a camera model its inner workings. For example, how the lens in the camera influences the image being captured. As already revealed in the previous section, the matrix describing intrinsic parameters of the camera (also known as the camera calibration matrix) in eq. (2) is rather simple and unrealistic. A more accurate version is shown below:

$$\mathtt{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

This is the actual matrix used in this thesis to represent the cameras internal parameters. The matrix shares a number of parameters also represented in the projection matrix used for actual graphics rendering. Exactly how the two matrices relate to one another and why this is of importance is described later in Section 8.

The paragraphs below detail each of the parameters in the matrix. When describing the parameters represented in the matrix, it is useful to consider the camera model containing a collection of rays projecting from the camera center through the image plane. Each ray can be considered to represent a single pixel. What the ray hits is eventually visualized on the image plane where the ray intersected with it.

Focal length (f_x, f_y) The focal length, f, is derived from the distance between the camera center and the image plane, as shown in Figure 2. The focal length of a camera affects the perceived perspective distortion in the resulting image. The focal length determines how close or far apart the rays projecting from the camera center are. If the focal length is very high, the rays are very close and the features of the picture will appear to be very flat and lacking in depth. If the focal length is very low, the rays will be far apart and the image will appear to have a very distorted perspective, because a short focal length, results in a very wide shot.

Note that the matrix shown in eq. (3) contains a focal length for both the x and y axis. This is because real cameras, in general, do not have the same focal length along each axis.

Skew (s) The skew parameter, s, helps model the rare case where the x- and y-axis are not perpendicular. The rays projecting from the camera center conform to the shape of the image plane. A skewed image plane will lead to skewed rays. Significant skew is very rare and the parameter is often just assumed to be 0.

Principal Point offset (p_x, p_y) Previously, the origin of the image plane coordinates was assumed to lie at the principal point. The origin is usually placed in one of the corners of the image plane. In this thesis, the origin was placed in the upper left hand corner. The principal point offset describes where the most "center" projection ray intersects with the image plane. In Figure 2, this would be the optical z-axis intersecting the center of the image plane. For real cameras, the principal point is usually located near the center of the image plane. In virtual cameras, it is often at the exact center.

An offset principal point not located in the center of the image plane, would mean that the rays projected from the camera center, all veer in the direction of the image plane, creating a shear.

Radial distortion Radial distortion is not commonly associated with the camera calibration matrix. This is because the intrinsic parameter matrix, neither describes, nor compensates for this distortion in any way. However, since radial distortion is an artifact created due to the inner workings of a real camera, I decided to include it among the other factors that can be compensated for.

So far, the camera model assumes that the projected rays emanating from the camera center are linear. That is to say, the camera center, the point on the image plane and the point in the world are all intersected by one straight line. In a real camera, using a lens, this is not the case. In actuality, the rays are bent when passing through the lens. The artifacts caused by this bending are easily spotted as straight lines are projected as being non-straight lines. Figure 3 shows images from two different cameras of a checkerboard pattern. There is a noticeable amount of radial distortion in both.

As already mentioned, radial distortion is not a part of the camera calibration matrix, is usually compensated for in post-processing.

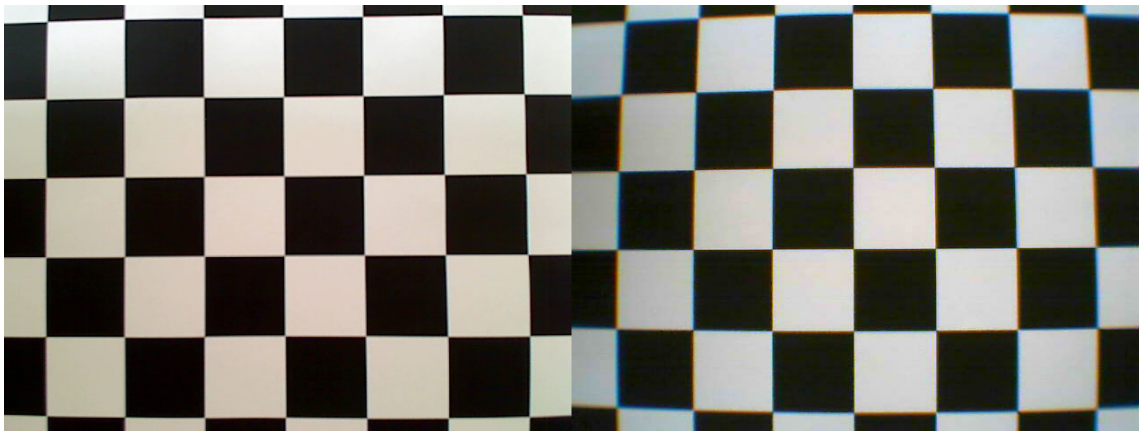

Figure 3: **Radial distortion.** Pictures from two different cameras of a checkerboard printed onto paper. Each image shows signs of radial distortion, the right one much more than the left one. The left image has been captured with a Logitech QuickCam Pro 9000. The right image comes from a Mini HiRes Webcam WB-3300p. Note that the printed checkerboard is slightly uneven, but as the images were taken from directly above the effect this has on the image is negligible.

5.3.2 Extrinsic Parameters

Just like the intrinsic parameters model the internal workings of a camera, the extrinsic parameters model the external workings of the camera. Specifically, the extrinsic parameters describe a cameras position and orientation in the world.

In the pinhole camera model shown earlier in Figure 2 the camera center is assumed to be located at the origin of the world coordinate system. In practice, this is usually not the case. It is preferable to be able to place and orient the camera in any way possible. Additionally, one may wish to use several cameras, and often one will want to place them in different locations. In fact, the camera center is often anywhere, but at the origin of the world coordinate frame. For mathematical and practical reasons it is preferable to permanently keep the camera center at the origin of the coordinate system. But we cannot move the camera around while simultaneously restricting it to the origin of the coordinate system. The solution therefore is to use multiple coordinate systems. One for each camera, and another for the world it must project. Assuming both coordinate systems are scaled equally and orthogonal, they are related via a rotation and a translation.

In order to project anything on to the image plane of the camera, it must share the same coordinate system as the camera. What is needed is an operation to bring points from one coordinate system into the other. Assuming our previously defined point $\mathbf{i}_{\mathrm{E}}$ resides in the (euclidean) world coordinate system and its counterpart representation $\mathbf{i}_{cam\mathrm{E}}$ resides in the (euclidean) camera coordinate system, then they are related by the following transformation:

$$\mathbf{i}_{cam\mathrm{E}} = \mathtt{R}(\mathbf{i}_{\mathrm{E}} - \mathbf{c}) \tag{4}$$

where $\mathtt{R}$ is a 3×3 rotation matrix representing the orientation of the camera coordinate frame (within which the image plane lies), and $\mathbf{c}$ is a 3×1 vector containing the camera center

expressed in world coordinates. The relation in eq. (4) can be expressed in homogeneous coordinates as

$$\mathbf{i}_{cam} = \begin{bmatrix} \mathtt{R} & -\mathtt{R}\mathbf{c} \\ 0 & 1 \end{bmatrix} \mathbf{i}$$

Given this formula, every point in world can be transformed to the camera coordinate system. What remains is to project them onto the image plane using the previously defined eq. (2). This leads to the following relation between a point $\mathbf{i}$, and its projection on the image plane $\mathbf{i}_{img}$

$$\mathbf{i}_{img} = \mathtt{K} \begin{bmatrix} \mathtt{I} & | & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R} & -\mathtt{R}\mathbf{c} \\ 0 & 1 \end{bmatrix} \mathbf{i}$$

which is also expressed in the matrix $\mathtt{P}$ commonly referred to as the **camera matrix**

$$\mathtt{P} = \mathtt{K} \begin{bmatrix} \mathtt{R} & | & \mathbf{t} \end{bmatrix} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{5}$$

where $\mathbf{t} = -\mathtt{R}\mathbf{c}$. The homogeneous counterpart to $\mathtt{P}$ is a simple extension of the matrices:

$$\begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Just like the matrix describing the intrinsic parameters, the matrix $\begin{bmatrix} \mathtt{R} & | & \mathbf{t} \end{bmatrix}$ shown in eq. (5) is closely related to a matrix used in graphics rendering. Specifically, the modelview matrix. Discrepancies between the two is explained later in Section 6.3.

6 Pose Estimation

The pose estimation algorithm presented and implemented in this thesis is based around homographies. According to Harley and Zisserman, given four or more points in the world, their corresponding positions in an image, as well as the internal parameters for the camera that produced the image, it is possible estimate the position and orientation of the camera in question [HZ04].

This pose estimation algorithm was chosen for two reasons. First, it performs pose estimation based on a single image, and can therefore be implemented using a single camera. This has the benefit of making the algorithm simpler than its multiple image counterparts. Second, it utilizes an analytic/geometric approach to pose estimation, as opposed to a learning based method. The analytic approach requires a predetermined amount of knowledge regarding the camera and/or its surroundings, in order to properly determine a pose. On the other hand, a learning based method is provided with a large set of images from which it can estimate the required parameters. The analytic/geometric approach is preferred since it relies on pre-determined parameters and in general leads to a simpler algorithm.

The following is a simplified version of the pose estimation algorithm:

1. Capture an image with the camera.
2. Identify a recognizable pattern (tag) in the image.
3. Extrapolate four corners and their coordinates from the recognized pattern.
4. Estimate a homography between the four extrapolated corners in the image and four corresponding virtual points.
5. Extract the cameras position and orientation from the homography.

Details regarding each step, from identifying a pattern in the image to estimating the position and orientation of the camera, is described in the following three subsections.

6.1 Tag Detection

Using the previously mentioned pose estimation technique, identifying four unique points is necessary. A number of the related projects [Kat, WAM$^+$, Fia, Lil03, Rek98, RA00] mentioned in Section 3, even ones using a different algorithm, have this requirement. A common solution to the problem is to use a very visible and identifiable pattern. Specifically, a recognizable pattern printed on a flat surface. This solution is especially appealing in this thesis, considering board games often have game pieces which must also be identifiable (for players). Exactly how this tag looks is irrelevant, as long as it can easily be recognized by both the computer and the players. An ideal choice is a square, considering it is the simplest of all geometric shapes made out of four points.

The tag shown in Figure 4 was chosen after a number of iterative design steps. The thick black border assures that the tag is easy to spot in as many places as possible when within the field of view of the camera. The size of the tag and the thickness of its design elements have been balanced to try and allow for the best detection while still allowing for as many tags as possible to fit within the view of any given camera. The end result was reached through imperative testing, and has yielded satisfactory results with the cameras used in this thesis.

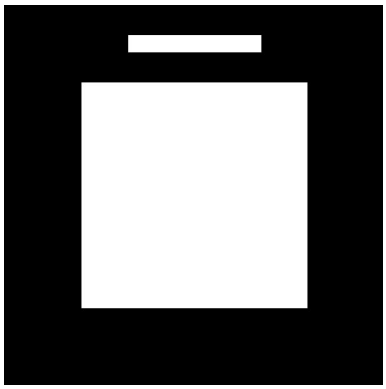

Figure 4: **A Tag.** The Tag used in this thesis for pose estimation. It is shown to scale and has a size of $5cm \times 5cm$. The top of the black border contains an orientation mark ($1,8cm \times 0,2cm$) and the middle of the tag ($3cm \times 3cm$) contains an ID code for unique identification.

Figure 5 shows how the original image captured by the camera is manipulated to identify any tags which may be within the field of view. The following is a complete list of the steps taken to detect any tags within the captured image:

1. The captured color image is corrected for radial distortion. The reduction in image distortion should lead to more easily detected patterns and and more correct perspective.

2. The corrected color image is converted to gray scale. This is necessary to perform thresholding on the image. Section 6.1.1 details necessary considerations regarding color-use for the recognizable pattern (tag).

3. The gray scale image is converted to a binary image, using the adaptive thresholding algorithm for the DigitalDesk [Wel93]. The same algorithm is also used in the cybercode tagging system [RA00], a predecessor of the technology used in the Eye Of Judgment [Wikd]. The algorithm contains a few adjustable parameters which have been slightly altered to better suit the detection needed in this thesis. The number of pixels considered in the thresholding history has been increased since the tags fill more foreground than individual letters. As the algorithm would scan a broad side of a potential tag, it would eventually declare it partially consist as background due to a relaxation threshold parameter. To avoid this problem, the relaxation threshold has been removed to improve detection.

4. Now that the image has been converted to a binary format, it is processed to detect contours. Every detected contour is approximated to a polygon which must satisfy the following criteria:

   (a) The polygon must consist of exactly 4 corners, otherwise it isn't a square. It must also comprise at least 1000 pixels in size to filter out any noise detected as a square. While it is possible for an actual tag to be contained within the space of a thousand pixels, it will be considered to be too small to detect properly. Finally, the polygon must be convex, just like the tag shown in Figure 4. As previously mentioned, a

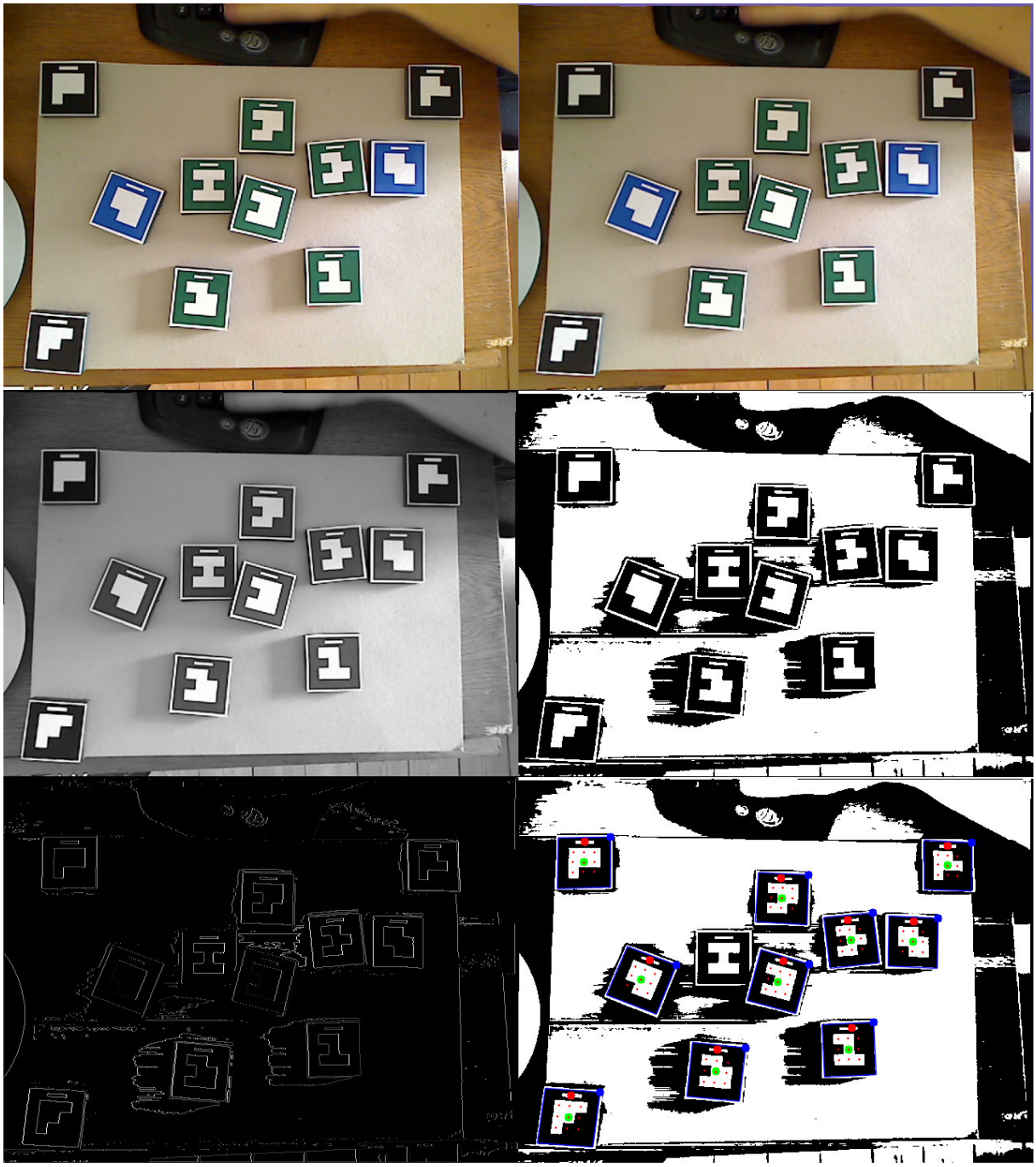

Figure 5: **The tag detection process.** Reading the images from left to right and top to bottom shows the individual steps taken by the algorithm to detect potential tags in the image. **Top-Left:** Original Webcam image. **Top-Right:** Radial distortion corrected image. **Middle-Left:** Gray scale image. **Middle-Right:** Binarized Image. **Bottom-Left:** Contour-detected image. **Bottom-Right:** Tag-Detected image. This image is mainly used for debugging purposes to give usable feedback on what the computer has recognized in original image. The blue borders signal a detected tag. The red dot indicates the center of the orientation mark. The green dot indicates the center of the tag. The blue dot marks the same top right corner of each tag. The tiny red dots show where the identification algorithm reads its pixel values.

projection will preserve the properties of straightness. Thus, the straight lines of a square, no matter the projection, will remain straight, and therefore it must be convex in the image to be considered a valid tag. All that remains is to make sure that the potential tag and its orientation mark satisfy the following criteria:

(b) First, the identification of an orientation mark is key to ensuring a tags authenticity. The polygon is checked for contours detected inside of it. Every internal contour is checked to be at least 25 pixels in size to filter out any noise. Finally, the ratio between the potential tag and the orientation mark ($tagsize/holesize$) must exceed a value of 40, but also be less than 140. Although the exact ratio between the size of the tag and the size of the orientation hole is known from a top down perspective, this ratio changes depending on how the tag is projected on to an image plane.

5. The four points of the estimated polygon are assumed to be a real authentic tag with which a pose can be calculated.

Initially, the algorithm had an additional step in between converting the image to a binary format and detecting contours in the image. To filter out noise, the mathematical morphology operations dilate and erode were applied to the binarized image. However, initial tests revealed that the operations did more harm than good and actually disrupted the application from detecting valid patterns. An example of this can be seen in Figure 6. In the top left of the image a dilate operation would break up the border of the a tag, rendering it unrecognizable.

6.1.1 Tag Color

A prominent feature of any pattern is its color (or lack thereof). Colors are often used to differentiate between similar looking objects. A good example is a traffic light. In addition to the individual position of each lamp, colors help to clarify what status the traffic light currently has. As previously stated, the exact shape and appearance of the tag is irrelevant as long as it can be easily recognized by the computer and the players. This can create a conflict of interest, as some patterns which are most easily recognized by players, may not be as recognizable for a computer and vice versa. For players, color are ideal to differentiate in between game elements. For computers, colors can be difficult to detect properly depending on the camera being used. Figure 7 shows two images taken with a low quality camera. Lack of lighting not only reduces contrast but also renders some colors almost indistinguishable from others. But even with proper lighting some colors are hard to detect.

Because precise color detection is neither easy, nor necessary, I have chosen not to use it when determining a tags proper identification.

However, the player must still be considered in this regard. Colors are ideal to help players uniquely identify game elements. Therefore, the tags should preferably be colored without interfering with the adaptive thresholding performed on the image. The adaptive thresholding technique [Wel93] mentioned in the tag detection algorithm was originally designed to detect text on a light background. In order to still use this technique it is important that the colors translate to a dark rather than light color value, when converting them to gray scale. Otherwise the tag risks being detected as background. Slightly altering the threshold values for the algorithm mentioned previously, and performing imperative testing showed that dark green and blue tags, shown in Figure 8, were ideal candidates. Originally red was considered as opposed to blue, but was discarded due to the prevalence of color blindness.

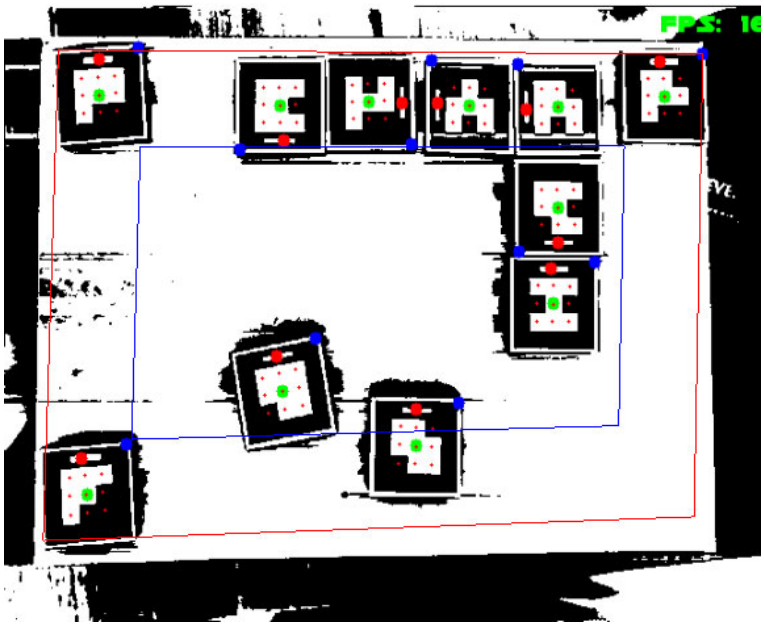

Figure 6: **Mathematical Morphology pattern disruption.** This is an image taken before it has been subjected to mathematical morphology. Pay attention to the five tags in the top right hand corner. Two of the tags show distinct white sections in the middle of the otherwise black borders. This is caused by the adaptive thresholding algorithms prolonged exposure to dark color. It eventually considers it to be background, and classifies further pixels thus. A dilution operation would break the tags borders and render it unrecognizable.

6.1.2 Tag Orientation

As previously revealed, the small white hole in the tag shown in Figure 4 is used to identify the orientation of the tag being detected. While it is not a strict necessity for successful pose estimation, it is useful to know from which side the camera is viewing a particular tag. As explained in the detection algorithm, the orientation mark is detected through a series of tests. Once it has been detected, it can be used to always uniquely identify the corners of the tag. Since the pose estimation algorithm calculates the correspondence between four points in an image and their counterparts in the world, it is vital to always associate the same points in the image and the world.

The method described in this section is just one of many possible methods to uniquely identifying the corners of a detected polygon. This approach was chosen because it is simple, works using a binary image, and also allows for human players to easily identify the orientation of a given tag.

The algorithm for uniquely identifying each tag corner is as follows:

1. Calculate the exact center of the tag and orientation mark.
2. Calculate and normalize the vector from the center of the tag to the orientation mark. This vector will be referred to as $dirVec$.
3. For each corner point, do the following:

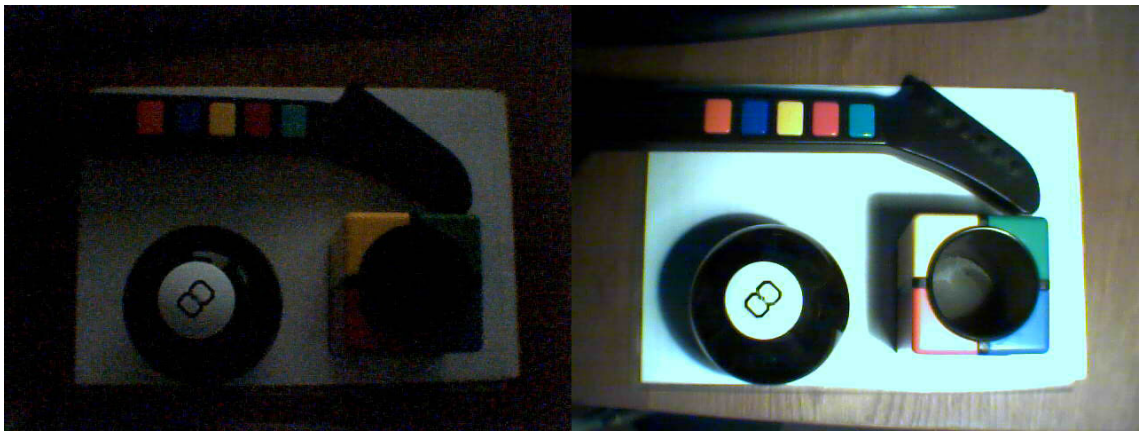

Figure 7: **Two images with different lighting.** In the poorly lit (left) image the different elements become hard to distinguish. The fourth color, from the right, at the top of the image, almost fades into the background, along with the colors the bottom right. In the brightly lit (right) image, most elements are clearly visible. Note that the top right square is actually green, although it appears as slightly blue in this light.

(a) Calculate and normalize the vector from the center of the tag to the corner. These will be denoted as $cornerVec_1$, $cornerVec_2$, $cornerVec_3$, and $cornerVec_4$.

(b) Calculate the signed angle α between the $dirVec$ and the $cornerVec_n$:

$$\alpha = \text{atan2}(cornerVec_n^{\perp} \cdot dirVec, cornerVec_n \cdot dirVec)$$

where the function "atan2(y, x)" performs an operation similar to the expression $\tan^{-1}(y/x)$ except that the signs of both quadrants are used to determined the quadrant of the result. The expression $aVec^{\perp} \cdot bVec$ is the notation for the "Perp Dot Product" and is calculated in the two dimensional case as follows

$$aVec_x * bVec_y - aVec_y * bVec_x$$

The signed angle reveals on which side of the orientation mark the corner is located as well as which of the two points on either side is closest to the orientation mark.

(c) Sort the points in the desired order.

Due to the nature of perspective projections, the algorithm described works given any projection on to the image plane as long as the tag and its elements are properly recognized.

6.1.3 Tag Identification

There are many conceivable situations where we would want to identify one tag from another. In most board games, players each have their individual pieces only they are allowed to interact with. Individually identifying each players elements is crucial in order for game play to proceed properly. The following approaches have been considered to individually identify tags:

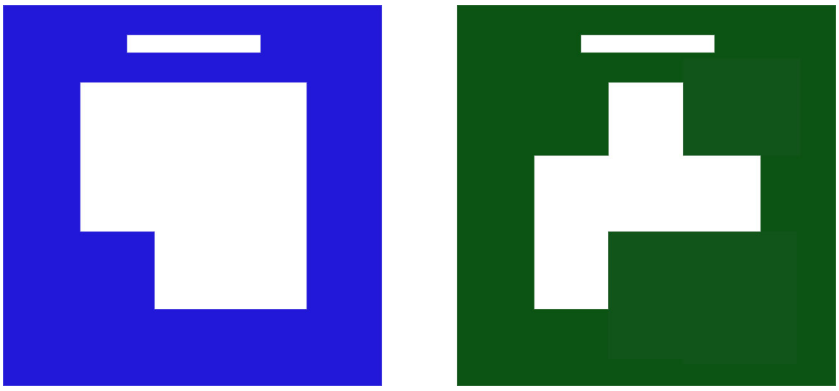

Figure 8: **Two differently colored tags.** Two tags shown to scale with the colors used in the final version of the tags. The actual tags used for testing were printed using the same printer as this thesis has been printed on, so the colors are near exact, ignoring any eventual printer color fluctuation.

- Identify each tag uniquely in every frame.
- Identify a tag uniquely upon its first appearance and estimate future positions based on movement.
- Combine the two above mentioned approaches.

I have chosen the first of the three possibilities as it is potentially the simplest of them. Continuously tracking elements based on their movement is complex and can cause many false positives unless properly implemented. Combining both approaches could potentially yield the best case solution, but is beyond the scope of this thesis.

As described in Section 6.1.1, I have chosen not to rely on colors when performing tag detection. The alternative is to detect a tag based on its shape. Most of the related projects which use tags, as a means of identifying four unique points in an image, also recognize a bit pattern integrated into the tag. The only exception is ARToolKit [Kat] which relies exclusively on a template pattern matching technique. To keep things as simple as possible, I have chosen to pursue a solution using bit patterns. The reader may have noticed that the tag shown in fig. 4 has an open white space in the middle. This is an ideal place to integrate a bit pattern into the tag.

Depending on how the tag is projected on to the image plane, it will become more or less distorted depending on the angle of the camera to the tag. Preferably, the bit pattern should be easily recognizable regardless of the distortion, and not require undistortion before it is identifiable. As previously stated in Section 1.2, the cameras exact position is unknown. The only guarantee is that the camera has the entire playing area are within the view. The tags themselves may have any position and orientation. The bit pattern should be as recognizable as possible from any given angle. Therefore, each bit to be identified should be allotted the same space in the pattern to give each an equally good chance of being detected when projected on to the image plane. In other words, the bit pattern must be completely symmetrical. I have chosen to use nine bits in the bit pattern code. This will allow for a

sufficient amount of unique tags (a total of 512 to be precise) and also yields a significant space for each bit to fill. Figure 9 shows a tag containing a randomly selected bit pattern.

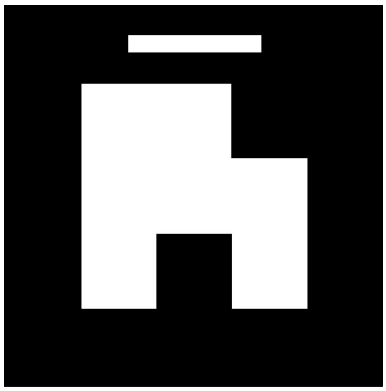

Figure 9: **A Tag with a randomly chosen bit pattern.** This particular tag has a value of 9, since the least significant bit (bit 0) is set as well as bit 3. $1 + 8 = 9$.

Knowing the exact dimensions of the tag and the location of its four corners in an image, it is straight forward to determine the location of the bits on the tag. Calculating a vector from both diagonally opposed corner pairs forms an X across the tag. How each bit is determined using these two vectors and their significance is shown in Figure 10.

Although a perspective projection distorts the size and shape of the tag, initial testing has shown that the large bit pattern compensates for the distortion.

6.2 Homography Estimation

The homography is an essential part of the pose estimation technique, used in this thesis. As previously described, the cameras position and orientation can be estimated from a homography. This section explains how the homography created.

There are a number of ways to estimate a homography, which describes a numerical relation between two sets of points. This section describes the simplest approach named the **Direct Linear Transformation (DLT) algorithm**. The homography to be calculated is a 3×3 matrix defined only up to scale:

$$\mathtt{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Since `H` contains nine elements, but is defined only up to scale, it is only the relation between the values which are of importance. Therefore, the matrix contains a total of eight degrees of freedom (dof) and a total of 4 unique points are required to constrain it. Each point - being defined with homogeneous coordinates - is also devoid of exact scale. Each point contains two degrees of freedom due to the relation in between the three coordinate values. Four points each with two degrees of freedom yield a total of 8 dof, and will therefore fully constrain the homography (`H`).

The homography `H` allows us to transform one given set of points into another, which yields the following equation: $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$. Where $\mathbf{x}'_i$ and $\mathbf{x}_i$ are homogeneous points in two

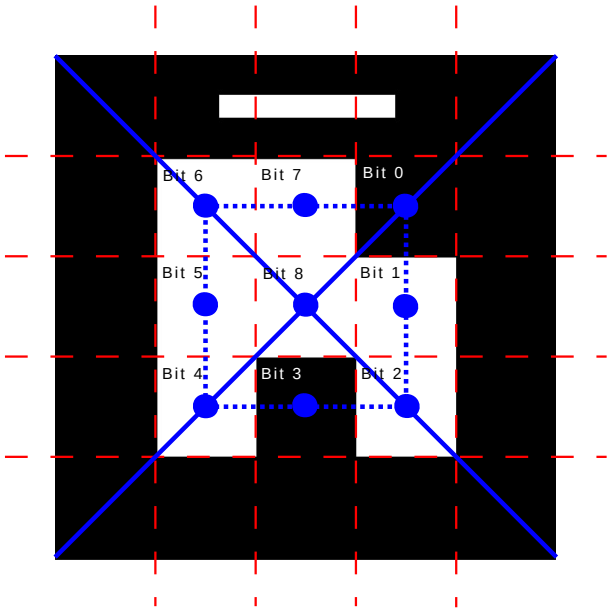

Figure 10: **A Tag and its identification pattern in detail.** This particular tag has a value of 9, since the least significant bit (bit 0) is set as well as bit 3. $1 + 8 = 9$.

different sets. It is important to note that since the equation involves homogeneous vectors, $\mathbf{x}'_i$ and $\mathtt{H}\mathbf{x}_i$ have the same "direction" but are not necessarily equal. Transforming the two points into euclidean space will however produce the same point for both points. In other words, $\mathbf{x}'_i$ and $\mathtt{H}\mathbf{x}_i$ may differ in a non-zero scale factor. When using the homography $\mathtt{H}$ in the above equation, $\mathbf{x}_i$ consists of the virtual point set while $\mathbf{x}'_i$ is the set of points from the image plane projection.

Since the homography $\mathtt{H}$ produces a point with the same "direction" as the corresponding point in the opposite set, the cross product can be used to express the following relation: $\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \mathbf{0}$, where $\mathbf{0}$ is a vector of zeroes. This equation can be reduced via the following steps. First, we denote the individual rows of the homography matrix $\mathtt{H}$ as $\mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3$ for the first, second and third row respectively. This allows us to express

$$\mathtt{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

By decomposing the point $\mathbf{x}'_i$ into its coordinate components (x'_i, y'_i, z'_i), the cross product equation mentioned above can be formulated more explicitly as

$$\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i - w'_i\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ w'_i\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i - x'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \\ x'_i\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i - y'_i\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

Because $\mathbf{h}^{j\mathsf{T}}\mathbf{x}_i = \mathbf{x}_i^{\mathsf{T}}\mathbf{h}^j$ for $j = 1, ..., 3$, the equation above can be expressed as a set of three equations in the entries of $\mathtt{H}$

$$\begin{bmatrix} \mathbf{0}^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \\ -y_i'\mathbf{x}_i^\mathsf{T} & x_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

Of the three equations, only two are linearly independent. The third row can be obtained (up to scale) by multiplying the first row by x_i' and the second row by y_i'. Discarding the third row yields the following set of equations:

$$\begin{bmatrix} \mathbf{0}^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

This set of equations can then be re-written and expanded to the following:

$$\begin{bmatrix} w_i'x_i & w_i'y_i & w_i'w_i & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i'w_i \\ 0 & 0 & 0 & w_i'x_i & w_i'y_i & w_i'w_i & -y_i'x_i & -y_i'y_i & -y_i'w_i \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

It is safe to assume that the homogeneous part of the coordinates of the points in this thesis are 1. Therefore, the set of equations can be further simplified to the following:

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

Each pair of point correspondences contribute with two unique equations. Given four pairs of points with no three being collinear, a total of eight unique equations are available. Stacking them together yields the following final equation:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 & -y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3'x_3 & -y_3'y_3 & -y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4'x_4 & -y_4'y_4 & -y_4' \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

With 9 unknowns and only 8 equations the system is underdetermined. But, as previously explained, we are satisfied with a solution up to an unknown scale. Therefore the set of equations is sufficient for calculating a homography for the purposes of this thesis.

6.3 Camera Position and Orientation Extraction

Now that the homography relating the virtual points to the points projected on to the image plane has been estimated, it's possible to extract the cameras position and orientation in relation to the tag. According to Hartley and Zisserman [HZ04], the matrix $\begin{bmatrix} \mathtt{R} & | & \mathbf{t} \end{bmatrix}$ first mentioned in section 5.3.2 can be computed from the matrix $\begin{bmatrix} \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_1 \times \mathbf{r}_2, \mathbf{t} \end{bmatrix}$, where

$$\begin{bmatrix} \mathbf{r}_1, \mathbf{r}_2, \mathbf{t} \end{bmatrix} = \pm \mathtt{K}^{-1}\mathtt{H} / ||\mathtt{K}^{-1}\mathtt{H}||$$

Where $\mathtt{K}^{-1}$ is the inverse of the camera projection matrix described in section 5.3.1 and $\mathtt{H}$ is the estimated homography described in the previous section.

The matrix $\left[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}\right]$ describes the relation between the virtual points and its related camera coordinates. The first two columns can be thought of as rotation vectors while the third describes a translation. But the matrix is not usable in its immediate form. Each of the columns contain a scaling factor. We want to maintain the distance and relation in between the virtual coordinates and only modify their distance from the camera. In other words, the tag the camera projects on to its image plane must not be allowed to change in size, but should instead variate its distance from the camera. Thus, we must extract the scaling factor from the two rotation columns and re-implement it in the translation column

$$\mathbf{t}_s = \frac{\mathbf{t}}{\frac{||\mathbf{r}_1|| + ||\mathbf{r}_2||}{2}}$$

Finally, we must normalize the rotation vector to make sure our final rotation matrix is not only orthogonal, but also orthonormal

$$\mathtt{R}_s = \left[||\mathbf{r}_1||, ||\mathbf{r}_2||, ||\mathbf{r}_1|| \times ||\mathbf{r}_2||\right]$$

We now have a 3x4 matrix describing the position and orientation (extrinsic parameters) of the camera viewing a tag:

$$\left[\mathtt{R}_s \quad | \quad \mathbf{t}_s\right]$$

This matrix is nearly identical to the modelview matrix commonly used in computer graphics. The key difference between the two, is the sign in front of the translation vector. The translation vector in the equation above, must have its sign inverted (by multiplying with -1), to convert it to the modelview matrix standard used in computer graphics.

7 Game Prototype

One of the goals set forth in this thesis is to develop a prototype game, which makes use of pose estimation. Preferably, the game should incorporate some elements from both board and computer games, so that it is uniquely suited for the type of interaction augmented reality allows for. The intentions behind creating a prototype game are two fold. First, to provide a proof of concept and test platform to evaluate how well the pose estimation functions in a real application. Second, to gain insight into how this type of tactile interface works when applied in a game.

Developing a fun and entertaining prototype game is not a straight forward matter. There is no tried and true approach to creating a game concept, which will entertain a select group of players, let alone players all around the world. A brief glance at most modern game titles today makes this apparent, as a very large portion of them are direct sequels, or at least reminiscent of previous successful titles. Even though many of the produced games are based on previously existing games, there are a number of tested and applied approaches to creating new game concepts and refining them into working prototypes [FSH04]. It should be noted that even creating the prototype, let alone the entire game, is not a short process and can take well up to a year or more, which is far beyond the scope of this thesis. Instead of creating a new game concept from scratch, I have chosen to examine a few existing ones and adapt one of them for use in an augmented reality application. Below I list some criteria, that the chosen and final adapted concept, should meet. They are to be considered general guidelines, and not strict rules. They serve as a personal guide, as to what I believe will yield the best result, in for a game using augmented reality.

- **Simple Game Concept** - There are several reasons as to why the adopted game concept should be simple by design. There has to be sufficient time to implement a prototype version of it. Testing a shallow game is more straightforward than testing a complex one. Determining proper cause and effect is easier, as less variables are involved. A simple game concept is also easier to learn (for the purposes of testing with players). An ideal game concept would be one that only requires one or two plays until a player has a firm grasp of it.

- **Minimal complex physical interaction between game elements** - This is a rather vague concept, and is best described by example. The European pick-up sticks game Mikado [Wikg] or Jenga [Wike] are two games which do not fit into the category of "minimal complex physical interaction between game elements". In these two games, the individual game pieces form a whole, and together form a working game concept. How the individual game pieces touch and pass force between each other, is crucial in both games.

  Games that do not fall into the category of "Minimal complex physical interaction between game elements" can still make use of augmented reality, but if the virtual parts of the game require similar physical interaction, the software must be able to support this in a stable and reliable fashion. Due to time constraints, such elements cannot be integrated in the software developed for this thesis. Therefore, game concepts containing or requiring such support are discarded.

- **Pre-defined closed environment** - As previously written in section 1.2, the software recognizing various game elements will only make use of a single camera focused on the

game area. As such, games that use a pre-defined and moderate amount of space are ideal. Most board games satisfy this criteria.

- **Active elements** - Something rarely found in board games, yet almost universally present in computer games. In my opinion, active elements make a game more engaging. I also believe that it is easier to focus a players attention if something is actively drawing it.

As previously mentioned in Section 3, a master thesis [Köf07] has developed a number of heuristics for the evaluation of the type of hybrid game, that is to be implemented in this thesis. It is beyond the scope of this thesis to fully evaluate the developed game concept, but the heuristics developed in the mentioned thesis have been taken into consideration, during the development of the prototype.

The following sections detail a number of game concepts that were considered as possible candidates for the final prototype game. Due to time constraints, only a single concept has been implemented as a complete prototype, to be tested.

7.1 Tower Defense

Tower defense [Wikm] is a computer game concept wherein the primary goal of the player is to stop the enemies (sometimes referred to as creeps) from crossing the terrain from one location to another. The enemies usually appear in waves (usually 20-30 enemies in a single wave), giving the player a short time to prepare in between each one. The player has a set of towers at her disposal, which she can place in a maze formation, in order to ensure that the enemies take the longest route possible, in order to reach their destination. Some towers are outfitted with weaponry, to wear down the enemies, and keep them from reaching their destination all together. Each tower is often associated with a certain cost, which the player must pay, in order to introduce it into the playing field. The player earns resources by dispatching incoming enemies resulting in both a positive and negative feedback loop. The better a player does, the more resources she is awarded to spend on even more powerful towers. If she is unable to cope with the advancing horde she is allotted a smaller amount of resources and, as a result, even less prepared for future foes. Advanced versions of the concept include additional management, in the form of supplying each tower with a sufficient amount of energy, or combining various forms of attacks to defeat ever more damage resistant foes. The player wins the game if the amount of creeps that slip through the defense is small (usually below 20-40). The player loses if too many creeps slips through (above 20-40), and must restart the entire game.

Although this game concept was partially implemented in a very rough version, it was eventually discarded due to the following reasons:

- **Game concept too complex** - Most versions of tower defense that I have encountered have a number of core mechanics in-common. One of them is the path finding ability of the artificial intelligence. It challenges the player to build some sort of maze, with which to slow the enemies progression through the terrain. Unfortunately, few other games resemble this style of play, and I believe that players would require a number of attempts before becoming proficient with this game mechanic. It is possible to discard this particular mechanic, but I fear that it would make the overall game concept too dull. Another issue with the tower defense game concept, is that of balance. How to

properly provide every player with a challenge is hard because there is an extensive amount of variables to modify. Game board size, nr. of tower available, tower damage, tower cost, enemy speed, enemy health, etc.

- **Lack of interactivity** - This shortcoming depends mostly on the player and her style of play. If the player is very skilled, she could potentially create the optimal maze and then sit back while the game unfolds with minimal interaction. Attempts to coax the player into further interaction would solve this problem, but would probably make the game concept more complex. Possible solutions to this issue includes forcing the player to upgrade her towers to stronger versions, or continuously changing the enemies entrance and exit positions forcing the maze to be reconstructed.

This was my initial choice for the game prototype, and I still believe that is would serve as a good augmented reality game concept. It was discarded due to its complexity and the time required, to properly implement a simple and balanced version of the game. The last recorded video of this game concepts short-lived prototype is included on the supplementary DVD.

7.2 Whack-a-Mole

Possibly the simplest of all the game concepts, Whack-a-Mole requires the player to counteract a number of moles appearing at random within the play area. To stop a mole from appearing the player must place a tag on the spot where the mole is about to appear. The challenge lies in how many moles the player can stop before her time runs out. It is the simplest of all the considered game concepts.

Although this game concept satisfies all the guidelines mentioned above, I decided to discard this game concept for the following reasons:

- **Game concept too simple** - The main concern regarding the simplicity of this game concept, is that the players might become disinterested before sufficient data could be collected.
- **Non existent element interaction** - Every game element is isolated from the others. Apart from the fact that one mole should not appear on top of another, the elements have no relation what so ever. I believe that this would lead to a less than ideal testing platform.

Although this game concepts shortcomings stems from its simplicity, it would probably be my second choice if there had been enough time to implement and test two prototypes.

7.3 Jigsaw Puzzle

An augmented reality jigsaw puzzle. Each tag represents a piece of the puzzle. Correctly assembling the entire puzzle finishes the game. The faster the puzzle is finished, the more points the player earns. A more complete version of the game could include a vast number of puzzles using standard image files formats. The concept could also be extended by giving the player a set of cubes to handle, each with a tag on every side. These six sides could provide the player with a bigger challenge by forcing her complete puzzles in more than two dimensions. This concept was only briefly considered and quickly discarded due to the following short comings:

- **Complex implementation** - In order for the game to work properly, a number of different jigsaw puzzles should be created. If only one is made, the players experience will be hampered by solving the same puzzle over and over. To avoid this, the implemented prototype should be able to automatically divide a given image into an appropriate number of jigsaw pieces for play. The game would also require an algorithm to detect whether or not these elements are correctly placed, in relation to one another.

- **Lack of active elements** - The game concept is very passive. The most dynamic part of the game is that the jigsaw puzzle can change after each level is complete.

7.4 Train Trax

The following is the game concept chosen for implementation as a prototype. As a result, it is described in much greater detail than the previous game concepts.

The concept of Train Trax is loosely based on an old game called Pipe Mania [Wikj], originally published in 1989. In the game, the player is in charge of placing an assortment of random pipes to guide a constant flow of liquid through a level. Every time the player places one pipe, a new random piece is made available. The player loses if the liquid running through the pipes reaches a pipe-less area. The player wins if she manages to guide the liquid through a required amount of pipes. In some cases the player is also required to finish the constructed pipes with a special end section.

The game concept behind Train Trax is similar, but instead of guiding a constant flow of liquid, the player is charged with the responsibility of guiding a constantly moving train. The player does this by placing train tracks in front of the train. The object of the game is to keep the train running for as long as possible, thereby earning the most points. A number of different tracks are at the players disposal. Three straights, three turns, one crossing and a random track. A single tag represents a single track. Which tag represents each of the provided tracks is randomly determined when the game commences. Figure 11 shows each track as it appears in the final version of the prototype game from a top-down perspective. Every track works in a straightforward manner, apart from the random one. The random track resembles the crossing track but sends the track in an unknown direction. The train can emerge in one of three directions: left, forward, or right. The train is only guaranteed never to return on the track it entered from.

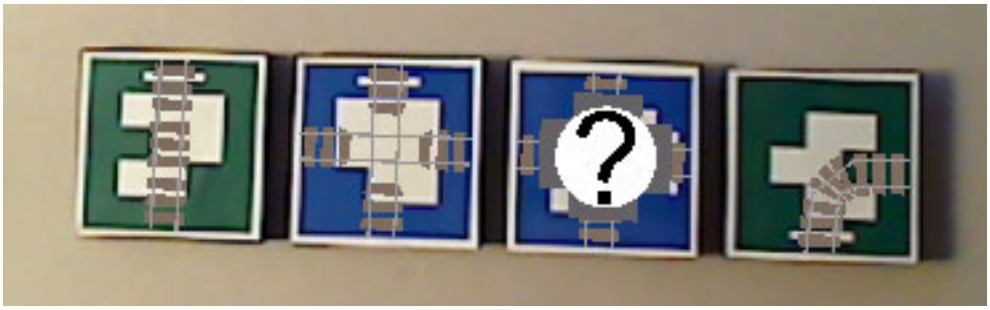

Figure 11: **The four different train tracks available from a top-down perspective.** From left to right the tracks are: a straight track, a crossing track, a random track, and a turn track.

The more difficult the track used, the more points the player earns. The crossing earns the player 50 points, the straight 100 points, the turn 200 points, and the random track 500 points. Figure 12 shows step-by-step play through of a single game session. One of the key differences between Train Trax and Pipe Mania is the fact that previously placed pieces can be picked up and placed in a new arbitrary position. In Pipe Mania, once a pipe had been placed and the liquid has started passing through, that piece could no longer be relocated.

A detailed step-by-step progression of a typical session is described below:

1. The application starts and the tags are placed in a haphazard fashion in front of the camera.

2. Three black tags indicating the playing field are placed at a distance from each other, and within the field of view of the camera. The size of the playing field only determines the possible spawning locations of the train station containing the train. Ideal positions for the tags are three corners of an underlying cardboard piece.

3. Once the three black tags are at their intended position, a fourth black tag, is placed in front of the camera to signal that the player wishes to start the game. The starter tag must be visible for a total of five seconds before play begins.

4. Once play begins, each tag reveals its associated track piece. The station containing the track spawns at a random location along the edge of the playing field, always facing toward the middle of it. The train station never spawns in either of the four corners. The player now has a short time available to place each tag in the intended position, before the train leaves the station.

5. When the train reaches the first player-placed track it slowly and continuously accelerates indefinitely. After a few seconds the spawned train station is removed from the playing field.

6. Eventually the train reaches a trackless area and the game is concluded.

Train trax was chosen as the final game concept because all of the previously explained criteria can be applied to it. However, it is not without its negatives sides as well. The most notable problems with the chosen game concept are listed below:

- **Complex implementation** - The most complex element concerning the game concept is undoubtedly the interaction in between the train and the tracks. Although the final design only includes four separate track pieces, they each provide a slightly different functionality. A relatively dynamic foundation must be built, upon which each of the four different types of tracks can be implemented.

- **Game concept too simple** - Compared to the Whack-a-Mole game concept, this is, in my opinion, far more advanced. However, the concept still lacks some elements to give it more depth. The only goal in the game is to accumulate as many points as humanly possible. Adding more goals would lead to a better player experience and more interesting game play.

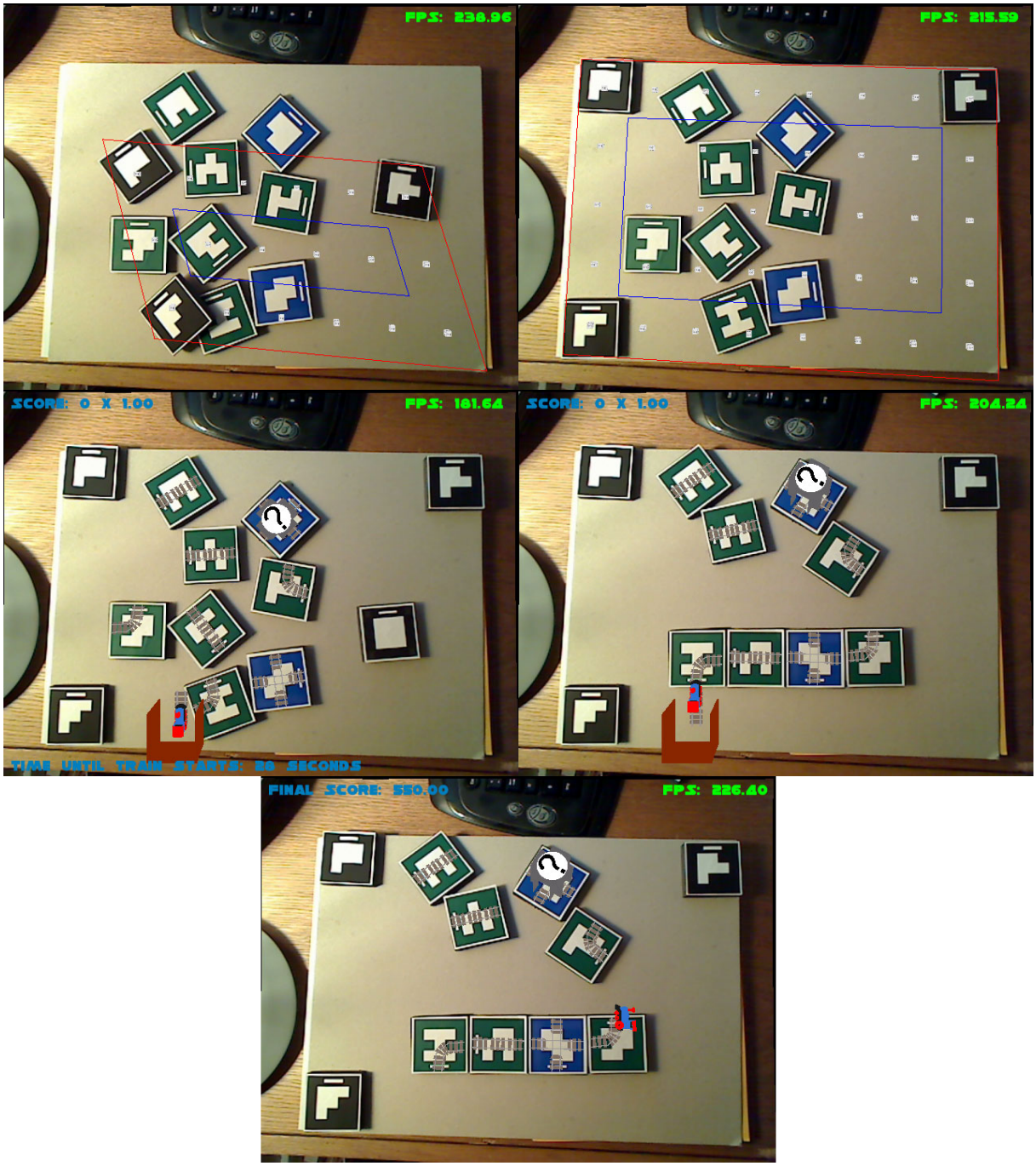

Figure 12: **A game played from start to finish.** Reading the images from right to left and top to bottom shows one game session in its entirety. **Top-Left:** The game starts with the tags usually placed haphazardly. **Top-Right:** Three black tags which designate the play area are placed in three separate corners to constrain and properly define it. **Middle-Left:** A fourth black tag, which signals the players intention to start the game, is placed. After five seconds within view of the camera, the game is started. A station, with the train on it, spawns at a random location along the edge of the playing field. **Middle-Right:** The player assembles the tags/tracks in their preferred fashion. The train leaves the station after a short while. **Bottom:** The train eventually reaches a track-less area of the playing field and the game ends.

8 Combining Realities

This section reveals how all the previously explained theory comes together, in order to track and visualize multiple virtual objects on different tags. To get a better understanding of what's been accomplished so far, we'll quickly recap what has been achieved in section 6. We have an algorithm capable of detecting potential tags in a given image as well the ability to calculate where the camera is located, in relation to the tag, that has been detected. We are close to being able to fuse the virtual reality together with the real one, but there are still a few barriers left to over come. Until now, the virtual points (of a tag) first mentioned in the beginning of section 6 have not been explicitly defined. They represent the virtual counterparts to the points detected in an image and should represent the shape of the tag. Thus, they must all lie on the same plane in a square formation. Theoretically, the points can be place anywhere in the coordinate system, but an ideal choice is around the origin of the coordinate system.

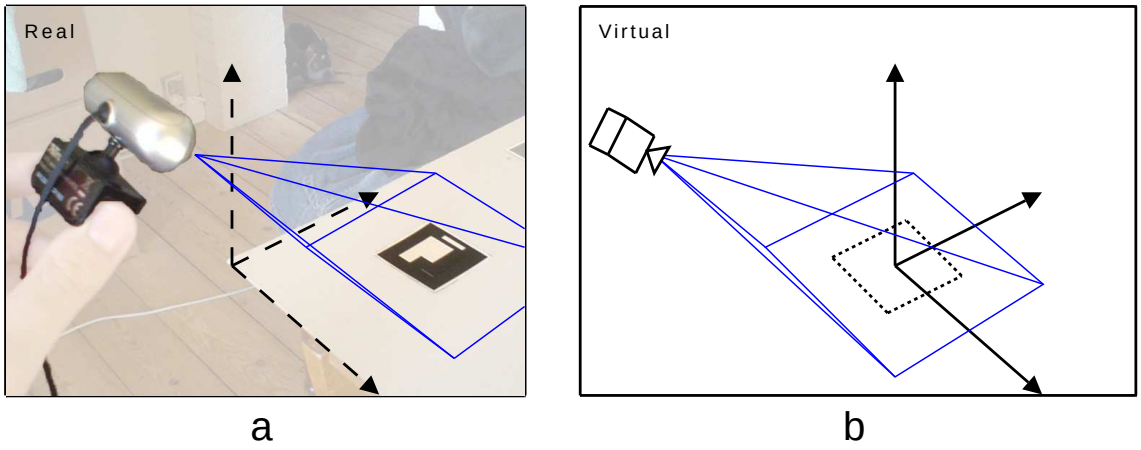

Figure 13: **Reality Vs. Virtuality. A single centered Tag.** (a) A camera observing a single tag in the real world. The coordinate axes are shown as a reference point. Obviously, the real world has no actual coordinate system. (b) The resulting virtual reconstruction. The virtual points are placed around the origin and the virtual cameras position and orientation result in a similar situation compared to the real world.

Figure 13 shows a comparison between the camera in the real world and the estimated camera in the virtual world. The tag is identified and the proper camera position and orientation is calculated. The actual coordinates chosen in the virtual world determines the size of the tag in relation to the virtual world coordinate system. I chose to make the tag unit sized in the virtual worlds which restricts the coordinates of the four points to the following:

$$\begin{bmatrix} 0,5 \\ 0,5 \\ 1 \end{bmatrix} \begin{bmatrix} -0,5 \\ 0,5 \\ 1 \end{bmatrix} \begin{bmatrix} -0,5 \\ -0,5 \\ 1 \end{bmatrix} \begin{bmatrix} 0,5 \\ -0,5 \\ 1 \end{bmatrix}$$

Figure 14 shows a situation similar to the one in Figure 13, but with an off-center tag. The real camera has been moved slightly so that the tag is positioned in the lower part, of

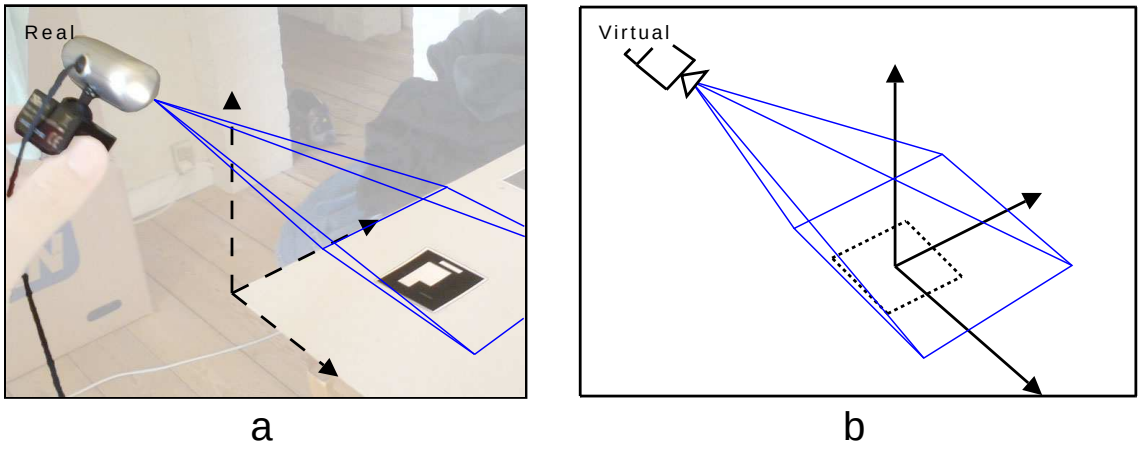

Figure 14: **Reality Vs. Virtuality. A single off-center Tag.** (a) The Real world situation. (b) The virtual counterpart to the real situation.

its field of view. The virtual camera is calculated to be positioned with a similar view of the virtual tag. As the real camera is moved or rotated in any fashion, the virtual camera will also be moved or rotated to the same position and orientation, as long as the tag is within the field of view of the real camera. In the situation just described (and shown in Figure 14), the tag remained static in both the real world and the virtual world. Only the camera moved and was estimated to a different virtual location. At first glance this seems reasonable, moving the real world camera causes the virtual camera to move as well. But this approach (of always re-estimating the cameras position) also presents a conundrum. How can we be certain that it is the camera that is moving and not the entire world? Using only a single camera, we cannot be certain. Regardless of whether the entire world is moving in one direction, or the camera is moving in the opposite, the situation will look exactly the same to the camera. Because the camera, in this case, is only "aware" of the tags it sees, moving the tag back and forth accomplishes the same effect as moving the camera forth and back. The pose estimation technique explained in section 6 always determines a new position and orientation for the camera, in relation to the tag. In other words, it is always the camera that moves and rotates to a new location in the virtual world. The virtual tag is always static, and the camera is dynamic.

The situation becomes slightly more complicated as soon as an additional tag is introduced into the scene. Previously, moving the virtual camera created a perfect match between the virtual and real world. Unfortunately, given two tags, two separate camera positions and orientations are produced. Figure 15 shows just such a situation. It is not surprising, given that both virtual versions of the tags, are assumed to be located at the origin of the coordinate system. In order for the virtual world to match with the real world, the two cameras must be relocated to the same location and have the same orientation. One could argue that the wisest approach would be to choose one camera to be static and relocate the other to its position and orientation. This would save the trouble of relocating and reorienting two (or more) cameras. However, neither of the two estimated cameras will ever be static as their location and orientation will be re-estimated whenever the camera or the corresponding tag

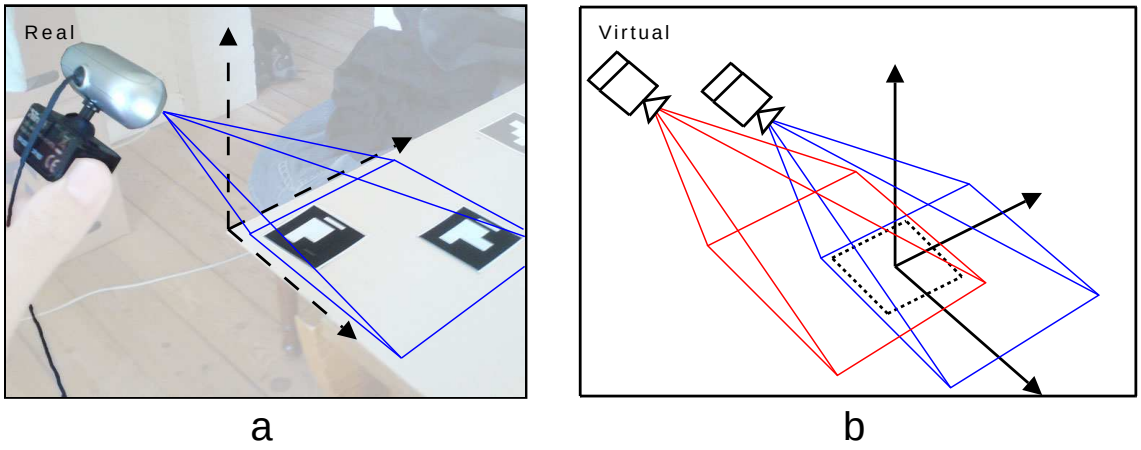

Figure 15: **Reality Vs. Virtuality. Two Tags.** (a) The Real world situation. (b) The virtual situation. A single virtual tag around the origo with two individual camera positions and orientations. This is problematic since we would prefer to have a single camera position and orientation with two individually positioned tags. This would allow us to render the virtual world on to a single window containing a video feed of the real camera.

moves in the real world. This could cause issues if virtual objects were introduced that did not depend on tags.

Instead, a fixed position and orientation is chosen and all camera positions are aligned to that specific location and rotation. It is vital that the virtual tag is also translated and rotated along with the corresponding camera, with the camera as its origin when rotating. Figure 16 is a visualization of how this process works. In fig. 16(a) we see the same situation as in Figure 15(b) except for the fact that there is now an additional (green) camera in the virtual scene. This additional camera is the permanently fixed position and orientation into which all other cameras will be translated and rotated. Figure 16(b) shows the first step in the process, which is translating all existing cameras to the permanently fixed position. As mentioned earlier, it is vital that the virtual tag each camera views is translated along with it. Only then will the tags be placed at the final proper position and realistically integrate with reality. The final step in the process is visualized in fig. 16(c). The two cameras are rotated into alignment with the fixed camera. Now the two estimated cameras are exactly aligned and will allow for multiple tags to be tracked using the same camera.

The tags in the real world and the virtual counterpart are now synchronized. However, even though the virtual tags properly correspond to their individual camera estimations, the virtual elements will not necessarily match up precisely with the real-world projection from the camera. The reason is that the two cameras (real and virtual) do not necessarily have the same intrinsic parameters. Even a slight difference in between the two cameras will propagate and become clearly noticeable when visualizing the virtual objects on the tags in the real world. Since its far easier to modify the properties of the virtual camera to fit those of the real one, that is what we will do. Given the intrinsic parameters of the real camera, it is possible to calculate the projection matrix as follows

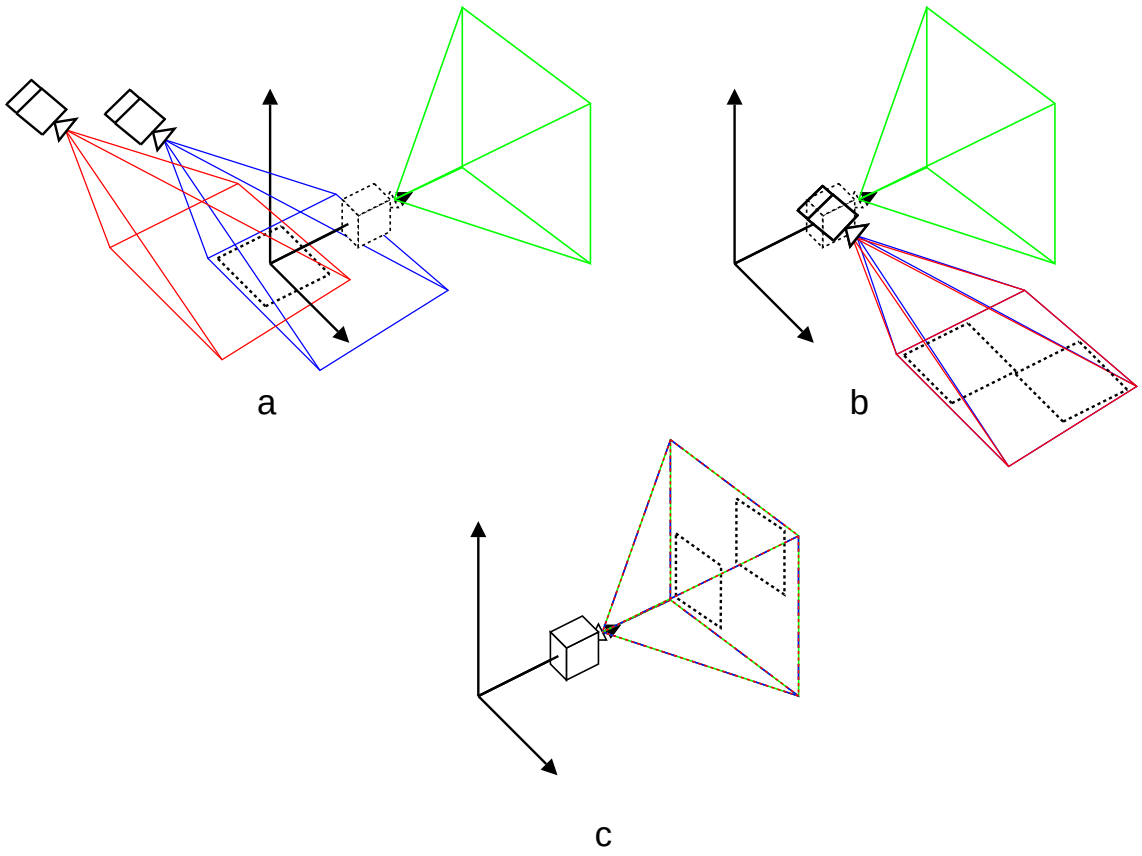

Figure 16: **Synchronizing all estimated cameras into one single position and orientation.** (a) A situation identical to the one in Figure 15. Two separate camera positions and orientations have been estimated. The dotted camera with the green projection lines is the static camera, to which the others must be translated and rotated. (b) Both cameras have been translated to the static cameras position. Notice that the two tags are now placed in a similar fashion to the real world in Figure 15. The projections match up nicely in this particular example, so the final step could be skipped in this instance. However, this is hardly likely to be the usual case, and an additional rotation is required, as shown in the next step. (c) The cameras are rotated to all face the same direction which aligns all the projections in the same direction. The cameras can now be treated as one, and all the tags are positioned properly in relation to it.

$$\mathtt{K}_{virtual} = \begin{bmatrix} \frac{2.0*f_x}{img_{width}} & 0 & \frac{2.0*p_x}{img_{width}} - 1 & 0 \\ 0 & \frac{2.0*f_y}{img_{height}} & \frac{2.0*p_y}{img_{height}} - 1 & 0 \\ 0 & 0 & -\frac{Plane_{far}+Plane_{near}}{Plane_{far}-Plane_{near}} & -2.0 * \frac{Plane_{far}*Plane_{near}}{Plane_{far}-Plane_{near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

8.1 Pose Estimation Stabilization

The pose estimation techniques described up until this point, will yield an acceptable result. The virtual objects will appear to be properly connected to the real tags that they match. Without further modification however, the resulting pose may at times appear unstable, especially if the tag in question is static. In such a situation the virtual object may appear to jump and jitter. The image quality, from which the tag is identified and the pose is estimated, also has a significant impact upon the visual result. In order to improve upon the pose estimation, an averaging algorithm has been implemented which is described in this section.

The overall goal of the algorithm is to improve the pose stability while keeping potential artifacts to a minimum. The averaging of multiple erratic values is a common solution to obtaining a more stable one. An averaging window containing five poses showed positive results and was chosen after initial testing. However, if the pose of a given tag is always averaged the virtual object will appear to be lagging behind the real tag. In an extreme case this could result in the user becoming uncertain of whether the two items are related. Therefore, the computer must be aware of when a tag is in motion and update the corresponding pose accordingly, without any averaging. In other words, the averaging should only take place if the tag is fairly static.

Initially, the estimated position of the tag in the virtual world was used as an indicator, for whether or not the tag in question had been relocated. Initial tests revealed that this value was too unreliable to depend upon. If the tag was located close to the camera, the position was estimated very accurately and fairly stable. But, if the tag was further away, the position became much more unstable. A static threshold for determining if a tag had moved, would yield poor results in either of the two situations. It is possible that a dynamic threshold, dependent upon the distance of the tag from the camera, would yield acceptable results, but a simpler approach was chosen instead. Testing showed that the image coordinates of the tag yielded stable results within a few meters distance from the camera. Therefore, a static threshold based on the euclidean distance between new and old point positions extrapolated from the image, should provide acceptable results. Using the four points that make up the outer board of the tag is ideal since they are readily available after the tag has been identified. The next appropriate question is how many of the points should be used. A sufficient number of points must be used to make sure, that it is impossible to rotate and/or move the tag, and still obtain similar coordinates for every point. Figure 17 shows a number of visual examples of how too few points can lead to improper poses. In Figure 17(a) only a single image coordinate is chosen. Clearly, the tag can be rotated and oriented in multiple ways without the coordinates of the chosen point changing. Figure 17(b) and (c) illustrates similar problems with using two separate corners. The final example (Figure 17(d)) is the only example where the coordinates of the corners is not exactly the same, in both poses. This example is included because it shows how a completely different pose can be achieved with relatively similar corner placement. In the left and right side of the example, the tag is facing to the left and right respectively. However, in contrast to the other examples, these two poses cannot be easily transitioned in between, without the computer taking notice of the change.

As a result, three individual points were chosen to determine if a tag remains stationary or had recently moved. If the tag was stationary, the algorithm would average subsequent poses. If the tag had recently moved the algorithm would discard any accumulated poses and start anew.

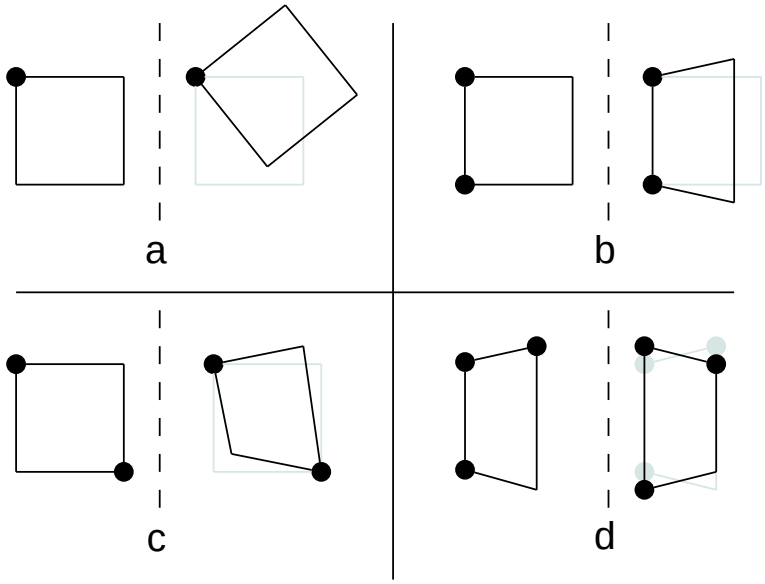

Figure 17: **Four examples showing how different tag positions and rotations yield the same positions for certain image coordinates.** (a) A single image coordinate is insufficient to evaluate if a tag has moved. (b) Two image coordinates prove equally problematic. (c) Diagonally opposed corners do not solve the problem. (d) Even three separate points can yield similar image coordinates for different tag poses.

For the benefit of the reader, the entire algorithm is described below in pseudo code:

1. Determine each of the three coordinates euclidean distance from averaged values of previous coordinates respectively.
2. If the sum of the euclidean distance for all three coordinates is higher than 10, the tag is determined to have moved and all previous averages are erased. The current value is used as the first out of five values.
3. If the tag has not moved and less than five averages have been collected, add the image coordinates, estimated pose, and estimated orientation to the average.
4. If more than five averages have been collected, discard the new pose and orientation.

9 Implementation

Despite the name of this section, the concepts explained within it only occasionally refer to the implementation specific details behind them. I believe the concepts are more easily described if the technical details are kept to a minimum. The code implemented during the thesis has been commented and contains a number of in-depth descriptions, if the curious reader wishes to know the exact details behind a given concept.

9.1 Virtual Board

Most board games include some sort of surface or mat upon which play takes place. Hence the term "board game". Likewise, most of the concepts described in Section 7 make use of a virtual board in one way or another. For example, in Tower Defense a common playing field is required for proper player and non-player elements to interact. Non-Player elements (such as enemies) must traverse the playing field and the game must be able to determine if and when a player has placed a tower tag within the field of play. Apart from being able to shoot the enemies, the tower must also be partially static during gameplay. By partially static, I mean that the player should not be allowed to constantly move a given tower within range of the enemies. It should be placed once and only allowed to move if the players resources allow it.

The only game concept which can easily do without a virtual board, is the Jigsaw Puzzle game concept. In that game, it is irrelevant where exactly each puzzle piece is placed, since every puzzle piece only relates to other puzzle pieces. If all of the puzzle pieces are assembled correctly in relation to each other, their position and orientation are irrelevant. The game concept Train Trax only uses a virtual board for the introduction of the train in the playing area.

To create a virtual board, it is necessary to define a section of a plane to represent it. Most board games use a rectangle shaped board to define the playing area. Since no accurate assumption can be made regarding the players surroundings and how the camera will relate to it, the player will be given the ability to define where the virtual board is located, and roughly what shape it should be. Each side of the virtual board will automatically mirror the opposite side of itself in length and angle. This allows for some very oddly shaped virtual boards, and it is left up to the player to place them in an acceptable fashion. Since two sides of the virtual board uniquely define its complete shape, it can be defined by three separate points. These three points are represented by three separate tags. Placing these tags in-front of the camera will define the position and shape of the virtual board. The three tags are pre-determined to be either the upper-right, upper-left or lower-right corner of the rectangle.

Once the grid border has been determined, the problem of how it should be represented internally arises. I decided that the chosen game concept was best served with a completely dynamic structure, where the player was unburdened by a grid or similar structure. However, a grid structure has been considered and implemented. In the final prototype it is only used to place the train station at the very beginning. Necessary considerations and details regarding this are explained in the following subsection.

9.1.1 Grid Structure

The simplest grid structure is comprised of equally sized square tiles. Square tiles would not necessarily fit well inside a user defined grid border, so the tiles are defined to be the shape of

the grid border. However, they are still all equally sized. A total of four different grid types have been implemented and are described below. Note that the term "tag-sized" only refers to an area the size of a tag, not a shape.

- **Manual Grid, Tiles are tag-sized** - The number of tiles within the grid is pre-determined, and the tiles are set to be unit sized, same as the tags.
- **Manual Grid, Auto-sized Tiles** - The number of tiles within the grid is pre-determined, and the tiles are re-sized to fit the placement of the three border tags.
- **Auto Grid, Tiles are tag-sized** - The number of tiles within the grid is determined by the amount of space within the user defined border. If there's room for another row or column of tag-sized tiles, it is automatically added.
- **Auto Grid, Auto-sized Tiles** - The number of tiles within the grid is determined by the amount of space within the user defined border. The size of the tiles are guaranteed to be at least unit sized. In other words, the amount of tiles along a row or column is determined by how many whole tags can fit along it. This is the grid type used in the prototype game.

Regardless of the grid type being used above, another issue remains to be dealt with. The issue of converting between virtual world coordinates and virtual grid coordinates. In most programs, grids are axis aligned. This saves complex computations when converting between the different coordinate systems. This approach is not possible since the player is in control of where the grid should be placed in the world. I have considered two different approaches to the problem. Either apply a transformation to the grid and all of its contents so that it becomes axis aligned, or separately determine the grid positions of elements, without affecting the grid. I have chosen to focus on the latter approach.

Converting from grid coordinates to world coordinates is straight-forward. The upper-left hand corner of the grid is defined to be it's origo $(0, 0)$. Two vectors spanning along the x-axis and y-axis of the grid from the origo are known. Normalizing these two vectors gives us $VecX_{\mathrm{W}}$ and $VecY_{\mathrm{W}}$. The subscript W denotes that these two vectors are using world coordinate values. Given these two vectors, the origo of the grid in world coordinates ($gOrigo_{\mathrm{W}}$), and the grid coordinates to be converted $(x_{\mathrm{G}}, y_{\mathrm{G}})$, the following formula will perform the conversion:

$$(x_{\mathrm{W}}, y_{\mathrm{W}}) = gOrigo_{\mathrm{W}} + (x_{\mathrm{G}} * VecX_{\mathrm{W}}, y_{\mathrm{G}} * VecY_{\mathrm{W}})$$

The formula will yield the upper-left corner of the desired tile in the grid.

Converting coordinates from world to grid is slightly more complicated. One approach is as follows: Create a vector ($pointVec$) from the grid origo to the point in world coordinates. Using the dot product, this vector can then be projected on to grid axis unit vectors and the exact grid position determined. The following equation performs the conversion:

$$(x_{\mathrm{G}}, y_{\mathrm{G}}) = \left(\frac{pointVecX_{\mathrm{W}} \cdot VecX_{\mathrm{W}}}{VecX_{\mathrm{W}} \cdot VecX_{\mathrm{W}}}, \frac{pointVecY_{\mathrm{W}} \cdot VecY_{\mathrm{W}}}{VecY_{\mathrm{W}} \cdot VecY_{\mathrm{W}}} \right)$$

9.2 Train Algorithm & Track Design

Section 7.4 briefly mentions that a dynamic foundation must be built with which each of the four different types of tracks can be implemented. This section describes the details behind that track system, and how the train interacts with it to stay on track (pun intended).

A number of different track systems were considered before settling on the final implementation described below. The design was chosen due to its simplicity and relative ease of implementation.

9.2.1 Track System

Each track consists of a set of track-connectors (also referred to as TCs from this point on). These TCs describe how the track can be connected externally with other tracks, as well as how a train is supposed to move when on it. Each track-connector contains the following information:

- **World position and Orientation** - This information is required so that the track can use this TC, to connect externally to other tracks. It is also required to place the train correctly in relation to the track, if it happens to be moving on it.
- **Next and previous TC** - This allows each TC to be internally connected other TCs. These two pointers only point to internal TCs, never to TCs on other tracks.
- **Next or previous TC is an arc** - This is essentially only used in the turn track. It reveals if the train should make an arc-like transition between two TCs, or if it should just commit to a straight line, which is the case, most of the time.
- **Center of arc position and orientation** - This information is only used if the train is traversing an arc. In that case, this position signals the center of the rotation, allowing for a radius of any size.
- **Foreign TC** - This information reveals if the TC is connected to a TC on a foreign track.
- **Score value** - Details the net-worth of crossing this TC.

Whenever a track is determined to have been moved, it searches within a short radius of its current location to find other foreign tracks. If foreign tracks are detected, each local TC then iteratively searches to see if any foreign TCs are close enough to establish a connection. If so, the connection is made, if not, any previous connections are removed.

9.2.2 Train Path Algorithm

Initially a number of fairly complex algorithms have been considered, able to find a complete path from the track beneath the train to the final connected piece. Apart from the fact that such algorithms pose a number of problems (for example, how to deal with a closed circuit), they are far beyond what is actually required.

Essentially, the train can remain fairly ignorant of how the entire track looks and needs only be aware of the track it is currently moving on and in rare cases a foreign track it is connected to. The train path algorithm is explained step-by-step below:

1. Based on the trains current speed, determine its distance to travel for the next frame to be displayed.

2. As long as the distance traveled by the train exceeds the distance left on the current track piece, skip train to next connected track piece. For each track skipped, deduct the tracks entire distance from the distance left to travel for the train.

3. If the distance left to travel does not exceed the total distance of the local track, the train must be placed somewhere along the current track.

4. For each TC on the current track, calculate the euclidean distance to the next TC. If the distance left to travel is below this value, place the train in between them correctly. If not, deduct the distance from the distance left to travel and move on to the next local TC.

5. If the train ever needs to leave the current track, and there is no foreign track connected to it, the train will stop and the game will end.

The above algorithm calculates the proper euclidean distance on tracks, weather they are straight or contains arcs. It also prohibits the player from extending the trains path by creating distances in-between tracks as the train will automatically jump from one track to the next. If the tracks are properly aligned, the switch can be virtually undetectable. However, if there is a significant gap, or the tracks are not properly aligned, the track will appear to "warp" from one position to another.

9.3 Tag tracking algorithm

How each tag is detected in every frame has already been explained in Section 6.1. However, the explanation in that section does not reveal how each tag is handled across multiple frames and objects related to detected tags are updated.

Every virtual object that is to be attached to a tag, registers it and is afterwards automatically synchronized to it every frame, with a given positional offset. Therefore, only one virtual object can be attached to any given tag. Tags are detected by finding contours in a binarized image, which are afterwards approximated to polygons. The detected contours exist in a tree structure, which can help reduce the amount of computations needed by not traversing the entire tree. For example, tags must at least contain an orientation mark which itself should produce a contour. Thus, a leaf of the contour tree cannot be a tag since it contains no contours. Initially, only the top three levels of the contour tree were analyzed for potential tags. Unfortunately, if the light was placed in a specific manner, a number of additional contours would arise around a potential tag, yielding the situation shown in Figure 18.

Consequently, an extensive number of contours are now checked before assuming that no more tags valid tags are detectable further down the tree.

After every tag has been detected, a final list comprised of all registered tags is checked, and if a registered tag has been detected, it is updated with a new location and orientation. If a registered tag has not been spotted in the past frame, it is also noted and eventually marked as inactive, after a total of five frames. As detailed in Section 6.1.3, each tag is uniquely identified every frame. In other words, a tags current location is determined by examining each potential tags identification. Although the system only supports registering

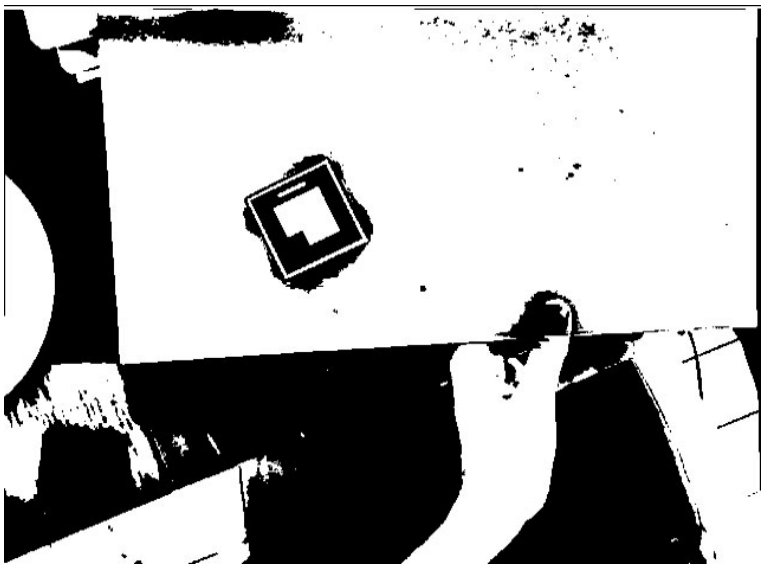

Figure 18: **A perfectly detectable tag, enclosed in a number of contours.** The first contour surrounds the entire image. The second is around the (now white) piece of cardboard the tag is placed upon. The third contour is the shadow surrounding the tag, and the fourth and fifth contours surround the actual tag.

each identification once, it can detect several tags with the same identification. If multiple tags with the same identification are detected, the last one to be detected is assumed to be the "correct" one.

10 Program Description

The application developed during this thesis has been written in the standard C++ programming language [Str] and uses the following external libraries and/or frameworks:

- **Object-Oriented Graphics Rendering Engine (Ogre3D) [SWJ+]** - Used as the primary graphics engine in the application. In addition, Ogre3D provides a number of tools and classes for working with objects in three dimensions, such as vectors and quaternions.
- **Open Computer Vision Library (OpenCV) [Con]** - OpenCV is used to perform a number of matrix and vector operations as well as edge/contour detection. Specific details regarding which portions of OpenCV is used is detailed in Section 11.1. In addition, OpenCV is also used to calculate a homography and as a comparison to the pose estimation used in this thesis.
- **Extremely Simple Capture (Escapi) [Kom]** - Escapi provides a very simple and straightforward API for initializing and using web-cameras via DirectShow [Mic]. The original source (kindly provided by the author) has been modified slightly to enforce a resolution output of 640×480 pixels and a maximum of 30 frames per second.
- **The Image Debugger (imdebug) [Bax]** - A programmers utility designed to make the debugging of windows applications, that use images, easier. The utility is executed along side the main application and can be provided with images in a number of formats.
- **Open Audio Library (OpenAL) [STC]** - A cross-platform audio library. A framework for using the OpenAL library is provided in their SDK which has been slightly modified and used in this thesis.
- **aStar Path-finding** - A module written by Jonas Drewsen which provides a* path-finding originally written for use in the Kryptonite Engine for SteamWinter [DBVL+]. The module was used in connection with the early stage development of the Tower Defense game concept (described in section 7.1), which was subsequently abandoned.
- **Simple Text Output (TextRenderer) [Aut]** - A simple class created for the purposes of producing simple on-screen text in the Ogre3D engine.
- **Paul Nettle's Memory Manager [Net]** - A memory manager intended to reveal potential memory leaks. Used sparingly in connection with the application and mostly out of interest of the author.

The implementation of the application in this thesis has been written from the ground up using Visual Studio 2008 Professional [Mic08]. Apart from the aforementioned libraries and/or frameworks, a camera calibration toolbox [Bou] was used to estimate the intrinsic parameters for two of the cameras used in the thesis.

Figure 19 shows a simplified UML diagram containing almost every class conceived for the application. For exact relations between the different classes, consult the source code provided on the supplemental DVD. Note that "classes" ending with an extension (such as `.h` or `.cpp`) refer to an actual file. As the diagram shows, the `main` function (contained in `main.cpp`) creates an instance of the `CamApp` class which then sets up the Ogre3D engine,

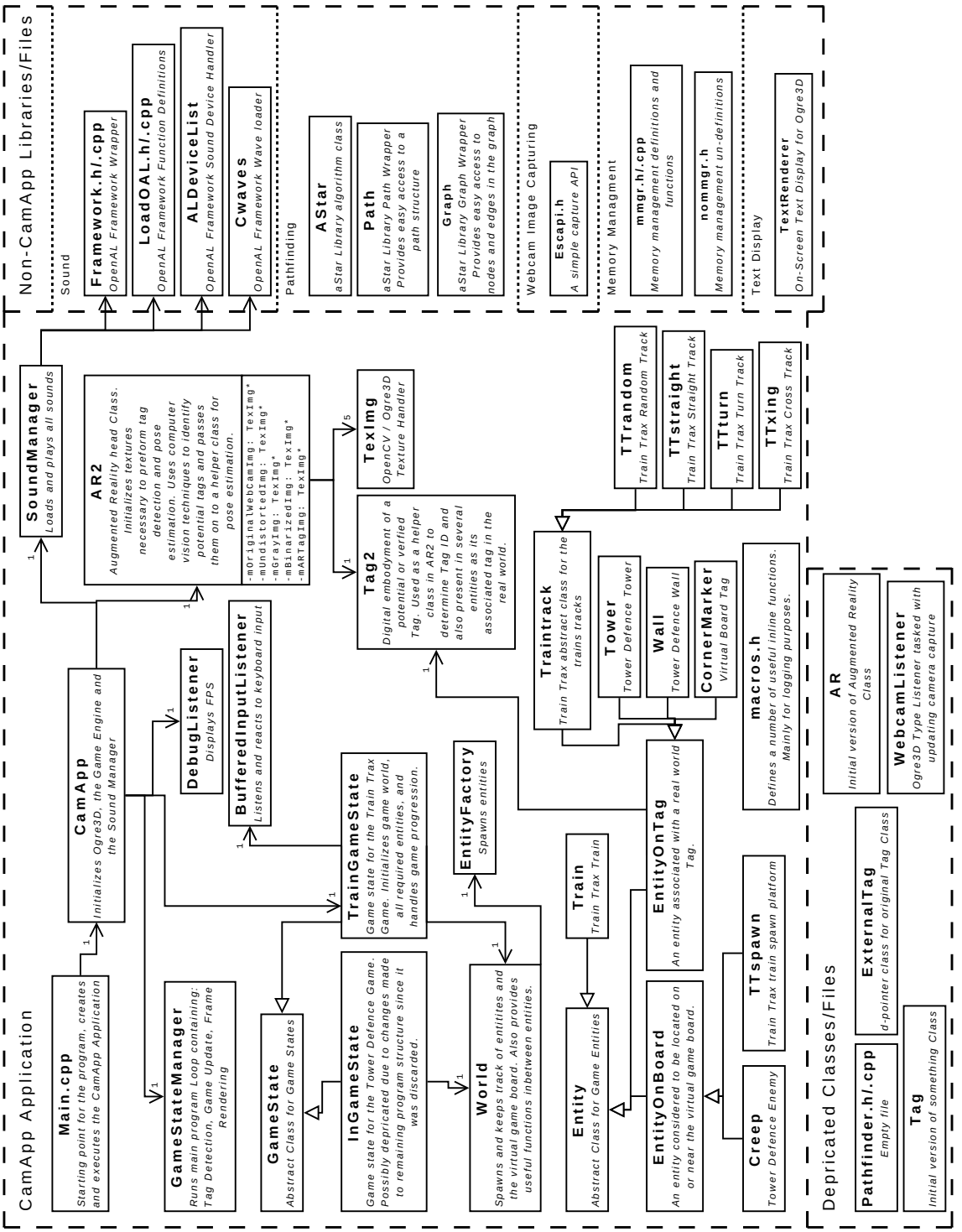

Figure 19: **A simplified UML class diagram of the program implementation.** The diagram shows a nearly complete list of all classes developed for the application. A number of omissions have been made in order to improve the diagrams clarity. Except for slight modifications to enhance output or improve integration with the main application, the external libraries have not been altered or created for this thesis.

DebugListener, SoundManager, AR2, and initializes the webcam before finally handing over control to the GameStateManager class.

Each of the classes created by the CamApp class perform a number of steps necessary prior to proper execution. The AR2 class creates its own private Tag2 object which it uses continually to determine the ID of each individually detected tag. The AR2 class also creates five instances of the TexImg class which handles the integration of textures between both OpenCV and Ogre3D, providing a seamless interfaces.

Prior to entering the main game loop in the GameStateManager class, the TrainGameState class creates the World class and the BufferedInputListener class. The World class is intended to be a multi purpose entity handler and is also used in the (now irrelevant) InGameState class. The BufferedInputListener class exists mainly for debugging purposes. Aside from interpreting the ESC button as quit, it provides no functionality intended to affect gameplay.

Once the main game loop is being executed by the GameStateManager class carries out five crucial actions: Determining the amount of time passed since the previous execution, capturing an image from the camera, evaluating the image for tags, updating the game state and finally rendering the entire scene. These five actions are repeated indefinitely until the application is finished. To evaluate the image received from the camera, the GameStateManager class utilizes the AR2 class. The World class handles the actual updating of the game world and every entity contained within as well as creating any necessary entities. It uses the EntityFactory class for this purpose.

The observant reader will notice in Figure 19 that there is an inheritance hierarchy with the Entity class at the very top. The Entity class provides a very basic set of functionality to integrate it properly into the Ogre3D engine. It provides a few accessors and creates a personal SceneNode which is one of Ogre3D's classes for visualizing graphics. The EntityOnBoard and EntityOnTag classes further specialize the entity to allow it to either seamlessly integrate with the virtual board (EntityOnBoard) or with a specific tag (EntityOnTag). The only Entity based class that does not rely on the virtual board or a tag, is the Train entity which continually shifts its dependence from one tag, to another.

The latest version of the source code, in its entirety, is included on the supplemental DVD, provided with the thesis. Additionally, a compiled binary of the code used during the user play sessions is included on the DVD. The compiled binary is located in the following folder: DVD/Source Code/bin/TestVersion/.

11 Testing and Results

This Section describes the testing procedure for the developed prototype and how the following two goals, set forth in this thesis, are fulfilled:

- Evaluate the robustness of the developed hybrid game system.
- Evaluate and analyze the users experience and interaction with the hybrid game.

To accommodate these goals, the robustness of the developed prototype is tested in two parts. First, the pose estimation and related algorithms are subjected to quantitative and qualitative tests. Note that only software implemented during the thesis, is considered, in this part. Second, the entire prototype is subjected to a black box test. In addition to the black box test, the prototype has been tested continuously during development. Significant discoveries, made during development, and subsequent changes, have been mentioned where relevant in the previous sections. The black box test is also performed in two separate stages. First, a pilot test is performed with a player to gain valuable insight, before the actual black box test proceeds. The pilot test is the last opportunity to discover and remedy any game design related problems, or implementation bugs. Once the final adjustments to the program and game design are complete, the second test portion can commence. A total of ten volunteer players are asked to play the game, preferably five times each, at least. This final test with ten players serves as both a more rigorous black box test of the implementation, as well as a platform upon which the players experience and interaction is evaluated and analyzed.

During the development of the prototype, a total of three different cameras have been been used. A W810i Sony Ericsson mobile phone camera [Eri], only used to obtain still photos during the initial testing of the adaptive thresholding algorithm. The video feed provided by the mobile phone is too poor in quality to be used for the live tracking of tags. The second camera, Mini HiRes Webcam WB-3300p [Tru], was used during a large portion of development and provides a video feed of sufficient quality, to test the functionality, of the pose estimation algorithm. However, the camera was subjectively determined to be incapable of providing a quality video feed, of a large enough play area, for acceptable pose estimation. Figure 3 in Section 5.3.1 provides a comparison shot between this camera and its successor. Each checkered square is approximately $3cm \times 3cm$ in size, which means the area in view is approximately $21cm \times 17cm$. This area would allow for a maximum of 4×3 tags which I believe would lead to a poor game play experience. Although the camera is capable of a much wider view, the blurring artifacts noticeable in Figure 3 only increase as the distance from the playing area grows. Therefore, it was replaced with the Logitech QuickCam Pro 9000 [Log]. As opposed to the previously mentioned models, the Logitech QuickCam Pro 9000 has a wide range of user adjustable features such as focus and exposure time. Without the ability to control these parameters, a substantial amount of test results, are left in the hands of the camera used to make them.

Ideally all cameras should be put through each of the upcoming testing scenarios to gain insight into their fundamental differences and effects on the end result. Unfortunately, such extensive testing is beyond the the scope of this thesis. Instead, a single camera will be chosen and used primarily during every test. The Logitech QuickCam Pro 9000 has been chosen since it provides manually adjustable settings, delivers the highest quality images, and potentially yields the best gameplay experience.

11.1 Internal Program Testing

This section details the evaluation of pose estimation and related algorithms. To get a better overview, the complete pose estimation algorithm is described step-by-step in the list below. Each step details the use of external software where appropriate.

1. **Capture an image.** - Uses a slightly modified Escapi library [Kom].
2. **Correct image for radial distortion.** - Implemented via OpenCV and is an optional step in the algorithm.
3. **Convert image to gray scale.** - Implemented via OpenCV.
4. **Binarize gray scale image.** - Implemented from the ground up based on the algorithm intended for use with the DigitalDesk [Wel93]. The new implementation relies on shifting pointers rather than shifting indices. The algorithm uses slightly modified parameters to better suit the needs of the prototype.
5. **Detect contours and approximate polygons.** - Implemented via OpenCV.
6. **Apply filters to ensure only valid tags are detected.** - Implemented from the ground up with some isolated OpenCV functions.
7. **Determine tag identification.** - Implemented from the ground up.
8. **Process all detected tags and update any registered by the prototype.** - This requires calculating a homography, estimating the cameras position and orientation from the homography, and aligning all estimated cameras and associated objects. Implemented from the ground up, apart from the homography calculation which is performed by OpenCV. Rotation and orientation calculations done in-part with Ogre3D [SWJ+].
9. **Average estimated poses.** - Implemented from the ground up and is an optional step in the algorithm.

As already noted in the previous section, only software implemented during the thesis is considered for testing, as testing all of the nine steps individually would be beyond the scope of this thesis. Therefore, steps 4, 6, 7, 8, and 9 are the main focus of the testing. The other steps contain functionality provided almost exclusively be external libraries and are assumed to function correctly. These steps will also be tested but only indirectly during the internal tests, and again as apart of the whole prototype during the black box tests.

The nine steps can roughly be divided into two separate categories. Identifying a tag and estimating that tags pose. Steps one through seven are considered to be in the category "Identifying a tag" and the remaining steps eight and nine are in the other category. The steps in the two categories are tested in the respective subsection below.

11.1.1 Tag Identification

The three main elements of the tag identification algorithm specifically written for this thesis are: the adaptive binarization (step 4), the tag filtering process (step 6), and the proper tag identification (step 7). In order to test the adaptive binarization, a qualitative analysis will

be performed where the prototype is subjected to a number of standard and extreme lighting scenarios. Although the extreme scenarios are highly unlikely to ever occur in a real life situation, their aim is to identify the limits of the adaptive binarization algorithm and the related causes. The testing will focus on situations where the camera is placed in a position intended for use in the developed prototype.

The tag filtering process is highly dependent on the results yielded by the adaptive binarization and will consequently be tested along side it. Tag identification is likewise dependent upon polygon yielded by the OpenCV library, which in turn is dependent upon the adaptive binarization technique. Its results will also be derived from the testing process performed with the adaptive binarization algorithm.

Whether or not a tag is identified correctly, can be determined by examining the visual debugging information provided by the system. The system shows a potential tag by drawing a blue border around it. If the tag is assumed to be an actual tag, additional information is displayed: a green dot indicating the tags center, a red dot indicating the center of the orientation mark, a blue dot indicating the tags primary corner, and nine small red dots indicating the bit pattern location. By observing the nine red dots and one blue dot, it is possible to confirm, if a tag is properly identified. If the blue dot is the first clockwise corner from the tags orientation mark, the order of the tags corners is correct. This means that the orientation of the nine red dots, used to read the tags identification, are properly oriented. If each of the nine red dots are located within each respective bit of the tag, then the tags identification will be determined correctly.

A total of 16 tags were placed in front of the camera at midday, sunset, and night. Within these three time periods the tags were subjected to various light conditions, such as indirect artificial light, direct artificial light and no artificial light. Additionally, the backgrounds were varied between a contrasting piece of cardboard and textured table top. Certain test scenarios were omitted due to overlap. For example, tests during midday, using no artificial light, and using indirect artificial light, fall into the same category since the sunlight fully illuminates the surroundings. All of the images from the test scenarios can be found in high quality on the supplementary DVD, provided with the thesis.

Figure 20 shows six separate sequences of indirect lighting during the three time periods on cardboard and regular table top. Out of the total of 96 tags in the six sequences, one is improperly detected. The improper detection occurs at night settings on a table top (the bottom sequence shown in Figure 20). The adaptive thresholding creates holes in the tag border, resulting in an improperly detected orientation mark. The cause is most likely contrasting dark shadow cast by the light on the lamps foot. The shadow causes the binarization threshold to assume the tag is partially background, leading to the artifacts shown in the image. This issue does not occur when using the cardboard, because it provides a sufficiently bright space in between the tag and the shadow. Note that in addition to the improperly detected tag, a four cornered polygon is detected in the same sequence and properly discarded due to the lack of an orientation mark. The same four cornered polygon is not detected in the first night sequence (third from the top) since its contour merges with that of the cardboard shadow. With the proper amount of indirect light, the different background appears to have a minimal effect on the overall detectability of the tags.

As previously mentioned, the camera and tags are also subjected to a number of extreme light scenarios. Although the tests were performed in all relevant time periods, only those with significant or abnormal results are shown and described below.

Because the adaptive thresholding algorithm determines which pixels are foreground and

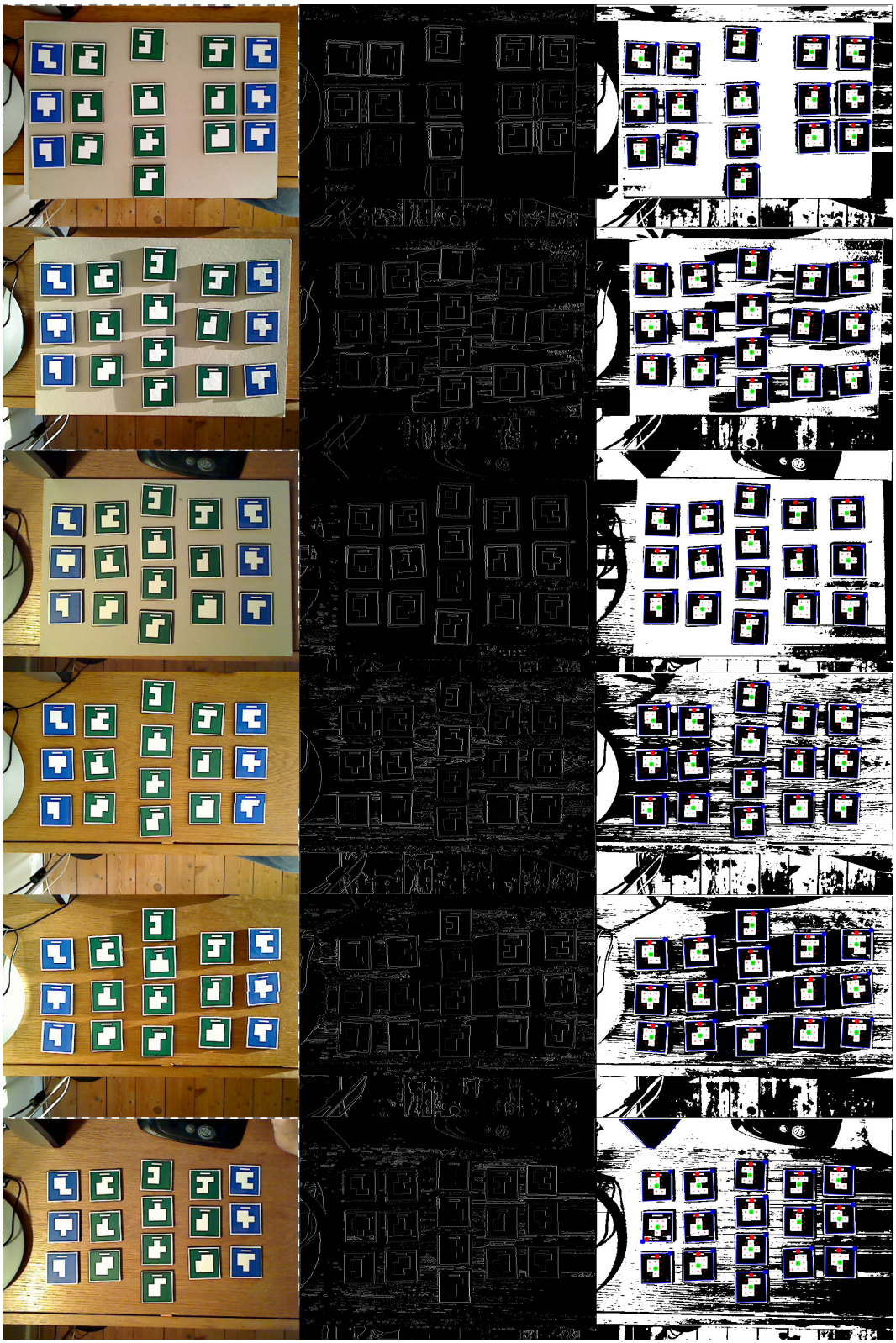

Figure 20: **Qualitative tests using indirect lighting.** - From left to right, each row represents the regular camera image, the detected contours and the resulting tag detection image. The top and bottom three rows are, from top to bottom, midday, sunset, and night respectively.

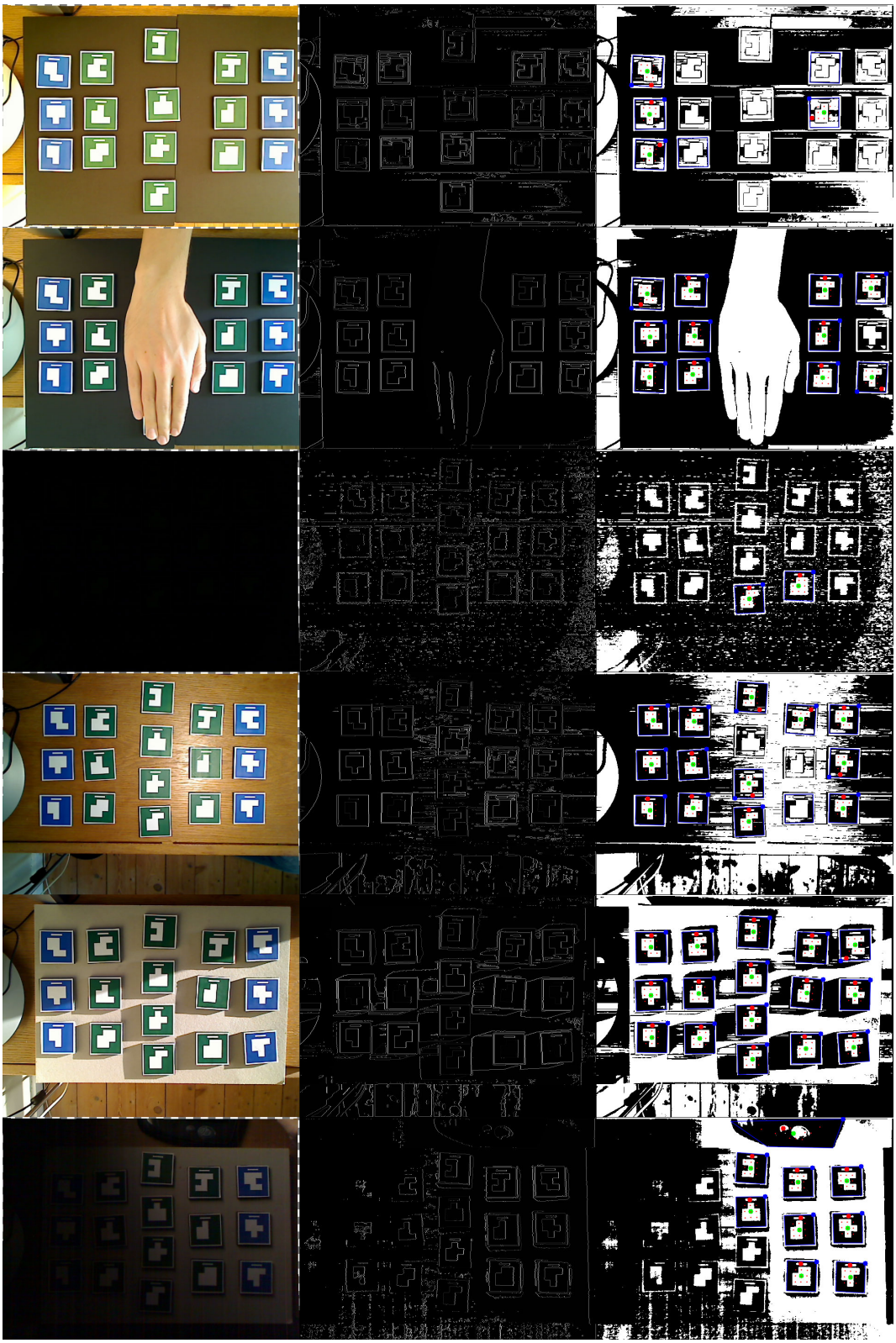

Figure 21: **Qualitative tests in extreme situations.** - The sequences are referred to from 1 to 6, top to bottom. The first four have been shot during midday, the fifth at sunset and the sixth at night.

background depending on the surroundings, a dark surrounding is likely to have a large impact on other dark elements. The first sequence in Figure 21 is a clear indication of this. Of the 16 tags in the image, only one is detected correctly, three are detected with an improper orientation mark, and the remaining 12 are not detected at all. Note that one of the tags with the improperly detected orientation mark, will still yield the correct identification and pose, since the blue dot is located at the proper corner. The poor detection in the image stems from the much darker color of the background, compared to the tags in the image. If the tags were of an equally or darker color, their chances of being detected would be better. Introducing a bright element which breaks the overall dark color of the image (such as a hand) results in a much improved detection ratio (shown in the second sequence in Figure 21). However, some tags are still improperly detected or undetected.

The third sequence in Figure 21 shows images shot at midday with the lowest exposure and gain settings on the camera. The actual image is essentially black to the naked eye, and consists of low `RGB` values spanning from zero to four. The contour image shows that even the slightest difference in `RGB` values will yield a separation between foreground and background, allowing for contours and consequently tags to be detected. Note that the poorer the signal (in the image), the higher the prevalence of noise in it. But despite the noise, two tags are still properly detected. In this type of light condition the success of a tag detection is almost entirely dependent upon how the noise is distributed, and if it breaks up a border beyond recognition.

Direct lighting onto the tags is shown in the fourth sequence in Figure 21. Direct lighting was tested during all time periods on both surfaces and the results all similar. The high concentration of light in one area, renders the directly illuminated tags unrecognizable, due to the bright light reflected back into the camera. Nine tags are unaffected by the artifacts created by the light, or have no artifacts at all. Although most of the tags outside of the light are properly detected, other tests have shown that tags further from the light source risk not being detected due low contrast caused by insufficient light. The sequence described was shot during midday and was the second to most forgiving direct light scenario. The scene most capable of handling indirect light, is during midday using the cardboard back. The cardboard back is a very matte surface and does not reflect light as brightly as the table top, and therefore causes less improper detection. Direct light at night creates an even stronger contrast in the image, rendering very few tags detectable.

The fifth sequence in Figure 21 is during sunset with half of the playing area in shadow and half in sunlight. As expected, the sharp contrast causes a few artifacts during thresholding. One tag out of the 16 is improperly detected, resulting in a reversed pose and incorrect identification.

The sixth, and final, sequence in Figure 21 shows images from night time, with poor lighting creating a gradual transition from dark to light across them. For a number of tags, the thresholding algorithm fails to create a proper outline, resulting in an undetected tag. However, the algorithm is capable of properly thresholding four tags which I can barely see on the image. This sequence is the only one to show a tag detection, where none is present. For reasons beyond me, OpenCV concludes that the oddly shaped blob on the top right is a square. The the tag filters detect a likely orientation mark, and calculate the "tag"'s identification. Fortunately, the resulting identification would not have matched with any of the tags in view of the camera, so in practice, this misdetection would not affect the program.

The thresholding algorithm performs well in evenly lit or evenly low-light situations, but is susceptible to a number of elements. Non-contrasting backgrounds, direct light and complete

image light gradients can all lead to undetected tags because of the adaptive thresholding algorithm. Using darker tag colors would assist the algorithm in differentiating the tags from the background, and therefore improve detection. The thresholding algorithm only takes a part of the image into consideration before evaluating if a pixel should be considered as background or foreground. By pre-processing the image to detect uneven light distribution, the algorithm could be improved. However, pre-processing an image every frame can be a costly procedure, computational wise. An alternative would be to fine tune the adaptive thresholding parameters mentioned in Section 6.1. A thorough quantitative test over a number scenarios with different parameters would likely help optimize them to best suit tag detection. However, I believe that very high or low contrasting images, such as dark backgrounds or direct light, may still cause artifacts.

The tag filtering process detects all tags in the squares discovered by the OpenCV library, including a single false positive. Because the tag filtering process does not discriminate between properly positioned orientation marks, it is no surprise that in certain occasions an artifact is detected as the orientation mark. By also forcing the orientation mark to be estimated to a square polygon or examining its position within the tag, such errors could be reduced. Note that polygon approximations of artifacts within tags, are often so small, that they will on a number of occasions also be estimated as squares. Given the few amount of non-tag squares detected by OpenCV, the filtering process warrants further testing in a non-game setting to further study its performance. Due to time constraints, this is beyond the scope of this thesis. In most of the tested instances, a properly detected and filtered tag lead to the proper identification of it. Only an improperly detected orientation mark shifted the bit order, yielding a false ID. The more rigorous orientation mark filtering mentioned above would help ensure that identification is properly determined every time.

Clearly, the thresholding algorithm should be a priority in further study, as proper filtering and tag orientation is irrelevant as long as the outline of the tag is not discovered.

11.1.2 Pose Estimation

This section describes how the pose estimation algorithm for this thesis has been tested. The pose estimation has been subjected to a quantitative comparison with the pose estimation provided by the OpenCV library. The comparison will provide a performance perspective between the two algorithms.

Determining the accuracy of an estimated pose is a complex task. I have considered two different approaches to measuring the accuracy of the pose estimation algorithm. The first is the conceptually most straight forward. It relies on the tried and tested method, of individually measuring each tags orientation and position, in relation to the camera, in both the real and virtual world. In the virtual world, these measurements can be performed quickly, with relative accuracy, and multiple times. In the real world however, the measurements are only as good as the tools used to measure them. In addition to being a very time consuming process, I doubt the accuracy of the measurements I would be able to achieve, given the current set of resources.

Instead, I have opted for a method which estimates results based on a known relation in between tags. A total of 16 tags have been placed on a flat surface (table top) from which a plane is estimated in the virtual world. Specifically, each of the 16 tags positions and poses are estimated, the values are averaged together, and a single plane is determined. The position and orientation of each tag is then individually compared, to this averaged plane, to measure

its deviation. A tags position is tested by determining the shortest distance between its position, and the averaged plane. Recall that one virtual world unit is equal to 5 centimeters in the real world. A tags orientation is tested by comparing its surface normal with that of the plane. The difference between the two normals is measured in degrees.

The pose estimation algorithm, detailed at the beginning of Section 11.1, contains two optional steps. Both the tags position and orientation has been tested with and without these steps, to see how they affect the overall accuracy.

In addition to testing on the table top, each measurement has also been performed on the piece of cardboard used during the user tests. Figure 22 shows the tags placement within the view of the camera. The blue tags have intentionally been positioned in the middle of the shot, to measure differences between centered tags, and those along the image border. In each test the surface was placed approximately 41 cm from the camera.

Figure 22: **Tag placement on two different surfaces.** The left image shows tag placement on the cardboard piece used during the game tests. The right image shows a similar tag placement, but on a table top surface. In both images the decimal value of each tag ID has been imposed to easily correlate its position and subsequent measurements in the following figures.

In each of the four figures displaying the results of the measurements, the x-axis is to be interpreted as an indicator for which test being performed. The following table shows which testing scenario relates to which x-coordinate:

X-Coordinate	Pose Algorithm	Pose Stabilization	Radial distortion compensation
1	OpenCV	OFF	ON
2	OpenCV	OFF	OFF
3	OpenCV	ON	ON
4	OpenCV	ON	OFF
5	Thesis	OFF	ON
6	Thesis	OFF	OFF
7	Thesis	ON	ON
8	Thesis	ON	OFF

Note that the tags have been slightly spread out along the x-axis to make the individual

tag measurement and variance easier to read.

For each of the eight tests shown in the four figures, a total of ten measurements were performed. These measurements have been averaged to a single value for each tag, including the maximum variance of the respective measurement. In all four figures, smaller values are better results. Perfectly estimated poses would show a distance and normal variation of close to zero centimeters and degrees respectively.

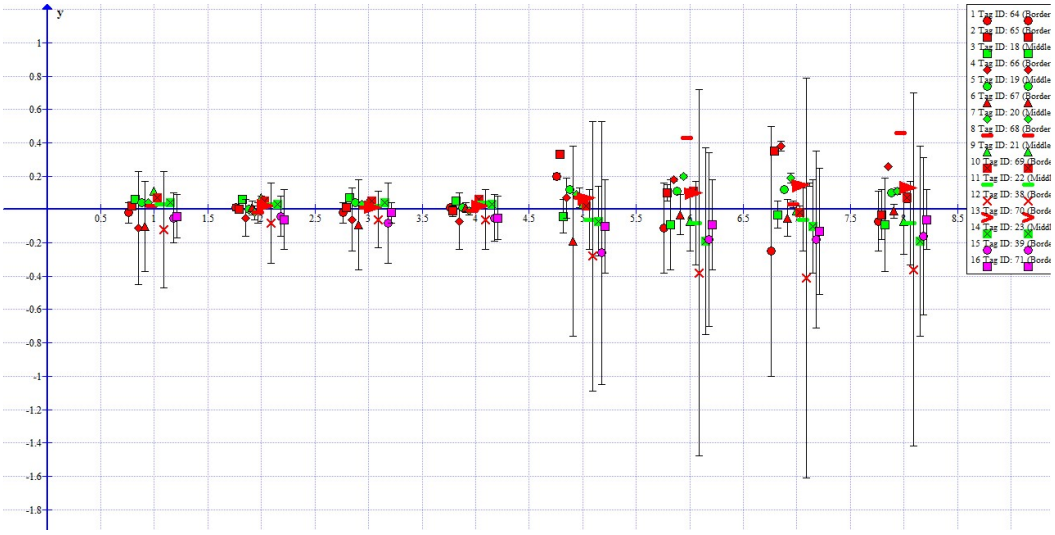

Figure 23: **Table-Top Position Comparison.** Green markers indicate centered tags. Red and purple markers indicate border tags. One unit on the y-axis equals 5 cm.

Figure 23 shows the eight positional test scenarios on the table top with 16 tags. It is immediately clear that the OpenCV pose estimation is superior in all four cases, with both a closer averaged position, and less overall variance. The tag causing the most variance using both the OpenCV and thesis pose estimation is nr. 12 with ID 38. It's worst measurement is 8 cm from the surface of the plane. The middle tags (shown in green) are in general closer to the plane than the border counterparts. Since the average position of the plane will lie among the center tags, this result is to be expected. The radial distortion compensation appears to worsen the overall results of the OpenCV algorithm, showing a higher variance in both situations where it is used. When using the thesis developed pose estimation algorithm, it actually appears to have a positive effect in the first of its four tests. Comparing the 4 situations where pose stabilization is on or off using both algorithms, the difference appears to be negligible. Considering that the tests already contain ten separate measurements, its not especially surprising, that the additional five, for each tag, make little difference. The best scenario showing the least deviation is the fourth.

Figure 24 shows the eight positional test scenarios on the cardboard piece with 16 tags. The most notable difference is that four last tests, most tags have a considerable variance, when compared to the last four table top scenarios. However, the few tags that caused extreme variance in the table top scenarios no longer deviate as much. Except for tag nr. 12 with ID 38, the OpenCV measurements are similar to the ones using the table top. It seems that, with the cardboard piece, the radial distortion compensation improves the pose estimation when using the OpenCV library, but mainly for tag nr. 12 with ID 38. However, it appears

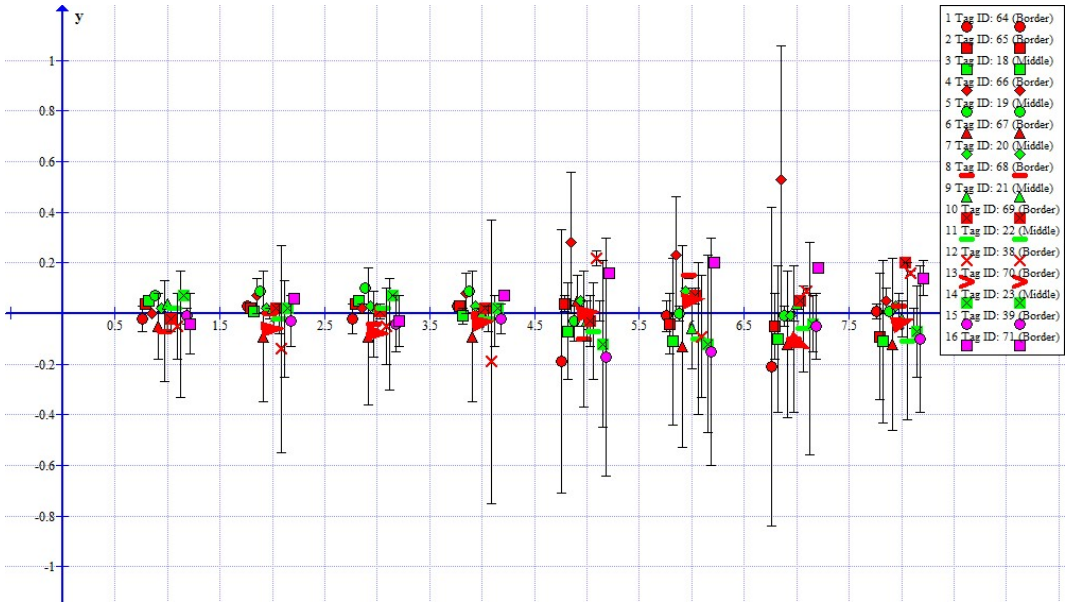

Figure 24: **Cardboard Position Comparison.** Green markers indicate centered tags. Red and purple markers indicate border tags. One unit on the y-axis equals 5 cm.

to have the opposite effect when using the thesis developed algorithm. The best scenario, with the least deviation, is either the first or third case.

Figure 25 shows the eight directional test scenarios on the table top with 16 tags. As with the positional measurements, the OpenCV pose estimation algorithm is more accurate. It's interesting to note that the tags with the most deviation, using the OpenCV algorithm, are the the ones in the center. This is something I also noticed during development of the thesis. Having the tag face the camera directly, often resulted in a jittery poses, rarely facing the camera directly. Using the thesis developed pose estimation, the center tags appear to deviate less than the border ones. The four tests with OpenCV library and thesis implementation each show an improving tendency towards the right. It appears that the best results are achieved using pose stabilization, and without radial distortion compensation. Although the last two tests using the thesis developed pose estimation appear to have tags deviating about the same amount. As with the positions estimated on the table top, the best scenario is the fourth one, using OpenCV using pose stabilization and no radial distortion compensation.

Figure 26 shows the eight directional test scenarios on the cardboard piece with 16 tags. In general, the eight scenarios appear to suffer from more deviation than in the table top tests. On the table top, the OpenCV algorithm had a maximum deviation of 8 degrees for a tag. Here it is approximately 13 degrees. On the table top, the thesis developed algorithm had a maximum deviation of 12 degrees for a tag. Here it is close to 15 degrees. Otherwise, the figure shows the similar improving tendency towards the right, as in the previous figure. Also similar to the previous figure, the OpenCV algorithm is more accurate overall, with the fourth being the best one. Measurements from the thesis developed pose estimation algorithm, show that most tags seem to deviate around 7-8 degrees, on the cardboard piece. On the table top, the degrees of deviation per tag is less centralized.

Taking all eight scenarios from all four figures into consideration, it is clear that the algo-

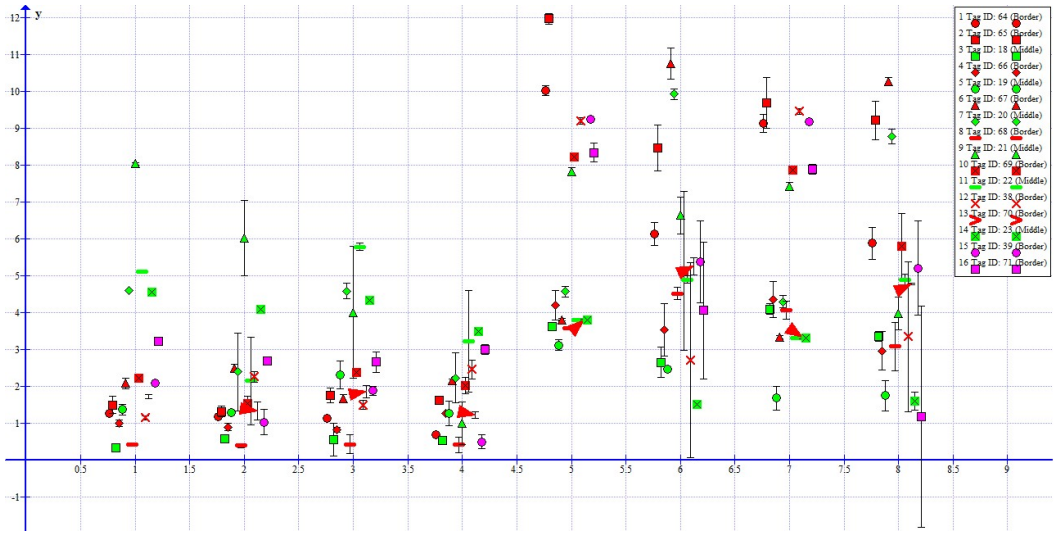

Figure 25: **Table-Top Normal Comparison.** Green markers indicate centered tags. Red and purple markers indicate border tags. One unit on the y-axis equals 1 degree of deviation.

rithm provided in the OpenCV library is the most accurate. A close analysis of the OpenCV source code reveals that the thesis developed algorithm is similar to the OpenCV in all aspects but one: Data normalization. Hartley and Zisserman [HZ04] show that data normalization significantly improve the calculated homography, when using direct linear transformation (explained in Section 6.2). Due to time constraints, data normalization has not been implemented in the algorithm developed for the thesis. I was surprised to find that in most cases the radial distortion compensation appeared to worsen the resulting pose, especially because the radial distortion, caused by the Logitech QuickCam Pro 9000, is minimal. This result could be caused by poor camera calibration. However, the calibration process includes a sample of no less than 20 different images which I would speculate yields fairly accurate results. Given more time, it could be interesting to perform the same tests using a camera suffering from severe image distortion and measure whether or not compensation improves the estimated poses. Section 11.4 details future improvements to the testing process.

11.2 User Test

This Section describes the black box testing of the complete prototype as well as the evaluation of the users experience and interaction with the game.

Preferably, all user testing should proceed in environment that the player is accustomed to [Mol94]. The equipment required to perform the testing is unfortunately too unwieldy, to carry in person, to one more more locations. Therefore, the testing was performed at an unfamiliar location, where every effort was made to put the player at ease. The testing setup, which was located inside a residential apartment, can be seen in Figure 27.

The testing setup was comprised of: one computer, one screen, a set of speakers, a camera, a microphone, and a IKEA-lamp (bought expressly for the purposes of this thesis, as a "camera holder"). The testing was concluded within eight days and the tests were performed between the hours of 09:00 and 17:00 to ensure the best possible lighting. The cardboard back was

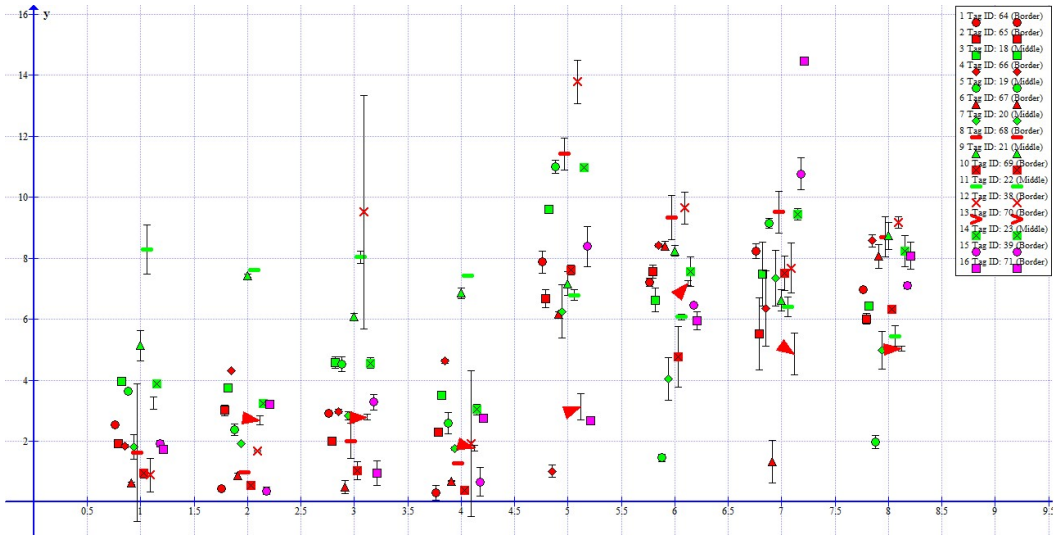

Figure 26: **Cardboard Normal Comparison.** Green markers indicate centered tags. Red and purple markers indicate border tags. One unit on the y-axis equals 1 degree of deviation.

used during user tests, as proper tag detection was deemed to be of high importance.

Prior to actual testing, a pilot test should be conducted in order to avoid any severe problems during the actual test [Mol94]. If a pilot test reveals too many problems, then an additional pilot test should be performed, after they have been remedied. A total of three pilot tests were performed prior to the first user test. The findings and corrections of the pilot tests are listed below:

- **Unreliable track connections** - A number of issues persisted with inter-track connections. Initially, tracks would only search for external tracks to connect to, if this was deemed necessary. This necessity mainly depended upon whether or not it, or other tracks, had been relocated. This approach resulted in unpredictable behavior, and was altered so that a track would continuously probe for surrounding tracks. Further complications arose due to these alterations, in connection with undetected tags. An undetected tag (and its virtual object), would always be removed from the playing area and subsequently disconnected from all surrounding tracks. Changes were made so that virtual objects belonging to undetected tags are never removed, only rendered invisible.

- **Train properties** - The speed and acceleration of the train were tweaked to better fit novice players. The initial speed was lowered and the acceleration increased.

- **Pose Stabilization** - The stabilization algorithm described in Section 8.1 proved problematic. Given an unfortunate circumstance, the pose estimation algorithm would sometimes estimate a very poor pose for a tag. Although four subsequent estimations are averaged into the equation, the original pose may be so poor that the end result, of the five averaged poses, would still be far from the correct pose. Since the averaging algorithm only re-estimated a pose after a tag had been moved sufficiently, the problem would never subside regardless of how long the prototype would recognize the tag. To

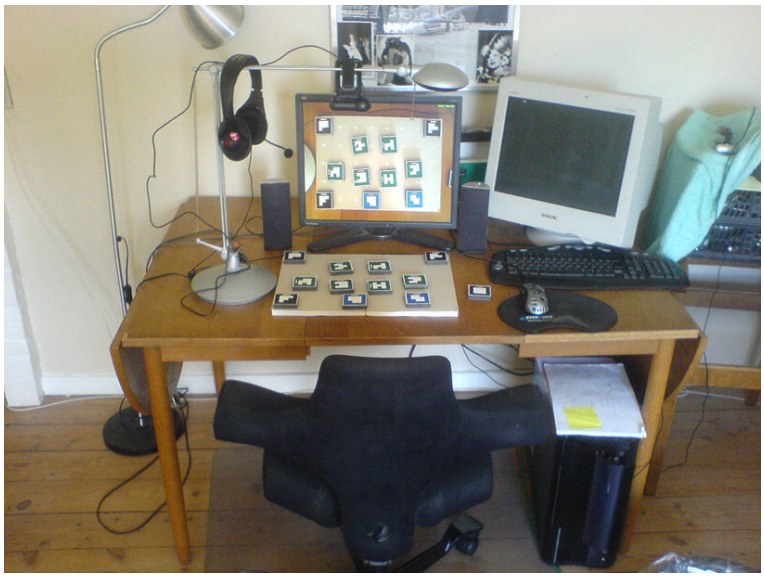

Figure 27: **Testing area.** - A cozy testing facility, complete with IKEA-lamp (using a Halogen bulb GY6.35 12V 50W). The IKEA lamp was used exclusively as a camera holder during user testing, and was never lit during user testing. In rare occasions, suffering from poor lighting due to cloudy skies, the lamp seen to the far right (using a standard 60W light bulb), was pointed towards the wall and turned on. This provided an adequate source of indirect light.

resolve the problem the averaging algorithm was modified to continuously update its averaging values.

11.2.1 User Test Procedure

The testing procedure used to test the developed prototype game, uses the "think-aloud" protocol where players are encouraged to voice their thoughts and opinions, during interaction with the application in question. This method of testing allows for a direct relation between application based events and associated user reactions. To get further feedback regarding the users experience, an interview is conducted with the player after the play session.

The developed prototype is intended for casual gamers. Ten players between the ages of 24 and 36 were chosen to participate. Prior to the actual test, each participant was informed by e-mail how the testing would proceed to minimize any anxiety they might have, in regards to the test. Once the participant arrived, the testing would proceed in the following fashion:

1. **Greeting** - The participant is greeted and thanked for participating in the test. She is asked if she is willing to be recorded during the testing procedure to improve data gathering. The participant is explained that it is the developed prototype that is to be tested, not the participant, and the entire procedure is explained in brief terms.

2. **Game Rules** - The prototype game and related rules are explained to the participant. Details regarding actual interaction methods are intentionally avoided. The player is made aware that she must obtain as many points as possible, by keeping the train

moving along train tracks. Each existing track is briefly explained and an association of "difficult tracks equals more points" is made clear.

3. **Game Prelude** - The participant is asked to relocated to the game area and is briefly explained which tags are merely present for functional use, and do not affect gameplay. The participant is asked to think out loud when playing the game and ask any questions they might have.

4. **Gameplay** - The participant plays the game a minimum of five times during which play and audio is recorded.

5. **Wrap-up** - The participant is asked a series of questions regarding the game and their experience, and finally thanked for their participation as the session is concluded.

Audio and video was recorded for each individual play session, and has been included on the supplemental DVD in a compressed format. Audio from each interview was also recorded, and has also been included on the supplemental DVD. An English translation of the questions used to guide the interview is provided in Appendix A. The original set of guideline questions are also included on the supplemental DVD.

The questions are aimed at getting an insight into the players perspective of the tactile interface and what issues it raised during gameplay. Certain questions are intentionally open and slightly vague to allow a broad interpretation and not limit the players initial response. In a similar effort, a few of the questions are repeated but phrased differently in order to gain as much knowledge regarding the subject as possible.

The results of the user tests have been divided up into two subsections. First, results relating to the black box testing of the program. This includes program instability, bugs or other implementation related issues. Second, results from user interaction and subsequent interview regarding their experience.

11.2.2 Black box Results

During the play-testing session, the prototype was played a total of 73 times. Each time the game session ended, the cause was recorded along with other statistics. Figure 28 shows all of the recorded sessions and the observed cause of their conclusion. Out of the 73 times the game was played, the session ended 25 times due to unwanted program behavior, 34 times due to the players behavior, and 13 times somewhere in between the two (designated gray area). Player behavior caused conclusions (or player "error"), involves a situation where the train ran out of tracks follow. Either due to a tag being placed too late, a tag being placed in time but facing the wrong way, or a tag being placed to far from the adjoining tag. Situations determined to be gray areas are one of the following: Train not properly transferring from one track to another while tags are in motion (pushed by player), or a tag being placed but never spotted by the camera, due to the player obscuring its view. Program "error" are situations where the train failed to transfer properly in between two properly placed static tags, or situations where one tag was improperly detected as another.

Out of the 25 times the game was concluded due to unwanted program behavior, 22 of them occurred when a train was switching in between two static tags, placed properly in relation to each other. The fact that they are placed properly next to each other is based on subjective observation, and therefore open for potential bias. However, the observations have

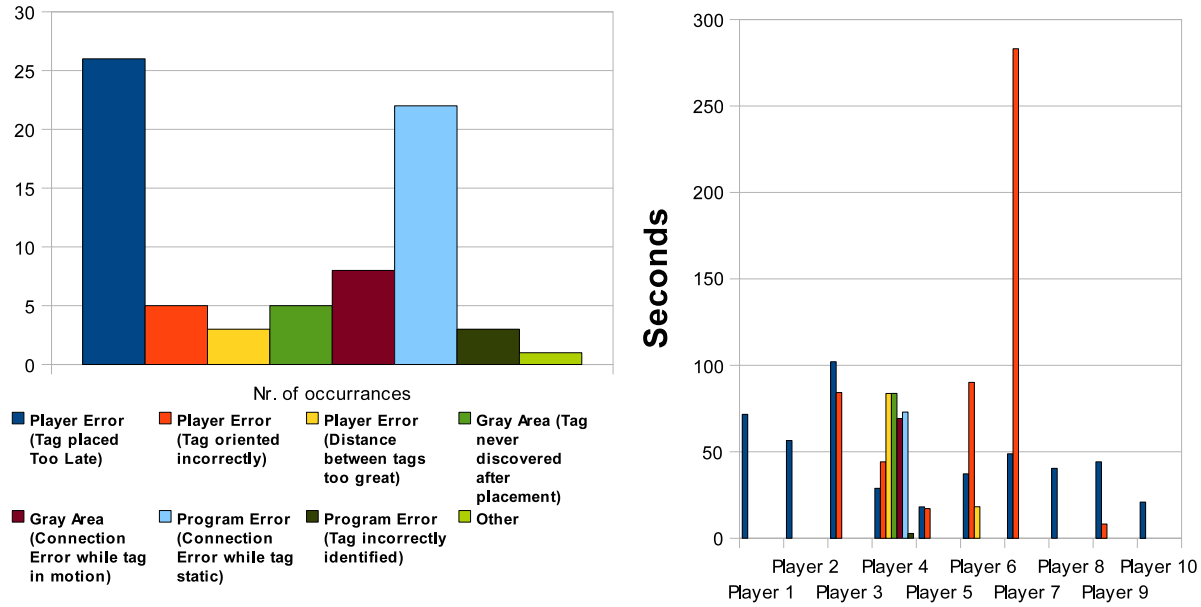

Figure 28: **Cause of game over and time statistics for unwanted program behavior.** - Shown left are the recorded game conclusions. Shown right are the times for all the game sessions which ended due to unwanted program behavior.

been checked several times and are in agreement with the respective players observation. The exact reason behind the improper transfer, of the train, from one track to another is unknown. Therefore, Figure 28 also shows a time distribution of these errors, for each player, in an attempt to recognize a pattern. Apart from the fact that most of the problems occur prior to 100 seconds, there appears to be no discernible pattern. Since this specific problem always occurs during the transfer of the train from one track to another, it is reasonable that given more time, more track transfers take place. The larger the amount of track transfers, the more chances for the problem to occur. The observant reader will note that player 4 experienced the most of these problems than any other player. No significantly different style of play was observed for that player, and I believe this is merely a coincidence. Based on the distribution shown in Figure 28, as well as the deviations measured in the prior sections, I assume that the unwanted behavior can be attributed to the pose estimation. If both tags, in between which the train is traveling, are estimated poorly, the tracks will fail to connect properly, leading to the observed problem.

The other unwanted behavior occurred a total of three times where one tag was mistaken for another. In every observed instance, this happened while the player was either moving or obscuring the tag in question. This, I believe, is also the cause to the misdetection. By performing a double check before accepting a tag is "detected", this behavior could be avoided.

Out of the 13 game conclusions in the gray area, 5 of them were due to a tag not being discovered after its relocation, due to the player obscuring it. This problem is not easily compensated for. Using the current detection algorithm even the slightest obscuring of the tag will often cause it not to be detected. A different algorithm is considered in Section 11.4 which could minimize the problem. How to properly handle issues caused by tags in motion

is less clear. Often multiple tags were moved by the player and preferably they should either all be detected, or all not be detected. Only detecting some of the tags could lead to track disconnections and cause an unexpected end to the game. Determining which of the tags in view are being moved along side others is no simple task and, if possible, would require advanced computer vision techniques.

The black box test clearly shows that the prototype exhibits some unwanted behavior. I believe most significant issues would be resolved by improving the pose estimation (by using data normalization for example), and verifying a tags existence before updating its position. Section 11.4 further elaborate on possible improvements. An unexpected conclusion to the game session is detrimental to the players experience. Resolving this program issue should prioritized in future work. Fortunately, out of the 73 game sessions, 34 ended exclusively due to the players behavior, providing ample data to analyze.

11.2.3 User Interaction and Experience Results

As previously described, the game sessions consisted of both a play portion and an interview portion. The following is a qualitative analysis of the observations made during the game sessions, as well as thoughts verbalized by the players during play, and their answers given during the interview. All of the recorded game sessions and interviews were analyzed and transcribed into a compact format after completion. The transcriptions have been evaluated and the most notable and reoccurring elements are detailed in this section. Videos of the game sessions and audio from the interviews are available on the supplementary DVD, in their entirety.

I have attempted to divide the input and behavior from players into two groups: during play and post-play (interview). Only where I found it necessary, are results from both intertwined due to a direct connection. The list below details the most notable observations and comments made, during the play session. Note that in the remainder of this chapter the word "tag" is interchangeable with the word "game piece". This is because players rarely refer to the tags as tags, but rather as game piece, which is a reference to both the tag and the cardboard base it is affixed to.

- **Quick familiarity** - The players were purposefully left in the dark, in regards to how the tags were related to tracks, and how their placement affected the game. Regardless, all players were able to assemble multiple tracks in their first attempt. Nine out of ten players also relocated tracks, that the train had already passed, in their first game session. I believe that this observation is an indication that players are quick to familiarize themselves with this kind of tactile interface. The word "indication" is purposefully chosen since a number of unmeasured factors can affect this player behavior, such as: previous gaming experience, general computer skills, or interest towards the prototype being tested.

- **Difference in focus** - Some players were observed to constantly shift focus between the screen and the playing area in-front of them. These players appeared to use the screen mainly as a temporary source of information, for the current train location and track placement. Other players began their session using the same approach, but eventually started focusing exclusively on the screen as a constant source of information, rarely looking directly at the tags in front of them. One player even commented, that he

did not notice one tag had been placed slightly on top of another, due to him focusing exclusively on the screen, and its top-down view.

- **Disappearing tracks** - Most players commented negatively, during their gaming sessions, on the fact that tracks disappeared, when the tags they were related to, were obscured. I believe that the players handled each tag as they would any other board game related element, which led to them being obscured from the cameras view. The tags were purposefully placed on 1 cm thick cardboard-like pieces for easier handling, but picking the tags up and not obscuring them is unwieldy and requires additional effort. Since the players were focused on playing the game, it is not surprising that the tracks were almost always obscured when picked up.

- **Real and Virtual World Separation** - All players appeared to instantly relate to train tracks being affixed to individual tags. However, every player that (usually accidentally) moved the train track carrying the train, was surprised to find that the train moved along with the track. I find this particularly interesting because in the real world, moving a train track with a train on it, would also move the train. A cause and effect scenario I believe most people would agree on. I believe the surprise that the players experience, is partially due to a distinction present in their mind, as to what is "real" and what is "virtual". The train tracks are affixed to real objects and therefore step closer to the "real" domain, than the train which appears out of no where, and cannot be physically interacted with.

- **Conceived grid in playing area** - On two separate occasions I noticed that players would place tags in a grid-like formation close to one of the three black tags. On both of these occasions, the game had just begun and the train station had spawned close to one of the three black tags. I think this is interesting because it indicates a reliance on real-world items, in an effort to satisfy a virtual goal. In other words, I believe that the black marker was temporarily seen as having a relation to the spawn point and therefore used as an anchor to coordinate the placement of the first train track. In both occasions the players eventually reevaluated the tags placement on-screen, and corrected the placement of the first tag. One player also explicitly commented in the interview, that it would make sense if the black tags were more directly related to the train station from which the train departs.

- **Additional feedback** - A number of players suggested additional visual and audial feedback during gameplay to help the player better interact with the game. For example, visual feedback showing whether or not tracks have been connected, or auditive feedback for when the train leaves the station.

- **Inner Workings** - Although the goal of the game was to achieve a high score by perpetually keeping the train moving, a few players started testing the prototypes limits after a few game sessions. Although this indicates an abundance of player interest towards the inner workings of the prototype, it is also a sign that the game concept had very limited appeal to these players.

- **Lack of clarity regarding the regular train crossing** - I was quite surprised that the regular train crossing caused some confusion for a few players. Prior to using the track the first time, a number of players were unsure as to how the train would react

when traveling across it. Most pondered whether or not it might stray to either sides instead of always going straight.

In addition to the observations and comments above, a number of players also expressed having fun while playing the game. Although this serves no specific purpose in relation to this thesis, it is encouraging to note nonetheless.

The list below details the most notable observations and comments during the interviews:

- **Tracks disappear when tags are occluded** - The absolute number one concern of all players, is the fact that train tracks disappear when the related tag is occluded. Every single player commented negatively on this particular issue, and most of them suggested that the tag itself should indicate what kind of track it represents. A few players also voiced concerns that the game might become too simple if one could tell what a tag represented by just looking at it. One player noted that the problem, was compounded by the fact, that the disappearing of tracks often happened, when the situation was most dire. The player referred to a situation where the train is almost out of tracks to run on, and the player has the least time to orient himself, regarding which tracks are where. In such a situation the need for clarity is highest, but so is quickly relocating the tags. I find this point especially interesting since it shows how the game mechanics are partially at odds with this kind of tactile interface. Constantly relocating tags also means that at least one train track is invisible, most of the time.
- **Game logic** - Overall, most players noted both during the play session and in the interview that the game was intuitive and the tags relation to the tracks logically fit. Players also noted elements which were less straight-forward:
  - **First Tag** - Placing the very first tag in relation to the train station from which the train departs proved difficult. This problem was also directly observed during game play a few times.
  - **Inter-track Train Transfer** - The immediate switch ("warping") of the train, between two tracks, gave some players an unnatural feel of the game.
  - **Train moves along with the track that it is on** - Just as observed during the play session, a player noted that this behavior was unexpected. The same player also added that he preferred this behavior, as opposed to the train losing its connection to the track, effectively ending the game.
  - **Game Start Timer** - A player commented on the timer sequence used to start the game. He explained that it was initially unclear as to whether the first five second count down signaled the trains departure (although the train is not visible at that point in the game). The same player also suggested replacing the timed train departure with a trigger system. Placing a tag in front of the train, should automatically trigger the system to begin.
  - **Scoring System** - How the score was affected by the use of each track piece was unclear to some players.
  - **Lacking Goal** - One player felt the lack of additional goals, made the game less logical.
  - **Black Tags** - A player voiced confusion regarding the black tags, and their relation to the game.

- **Tactile interface as part of the game** - Most players noted that they perceived the tags as being an integral part of the game. Any difficulty experienced with moving and coordinating tags was often attributed to being a part of the games challenge. A few players noted that they appreciated this type of interface, as opposed to alternatives, such as a mouse or keyboard. One player explained that the greatest strength of the interface was controlling the rotation of an object. He personally found this extremely intuitive compared to alternatives provided by other types of interfaces (using the mouse or keyboard).

- **Proposed changes** - When asked, the players suggested a number of changes. I have limited the suggestions here to those most relevant to the interaction and game concept:

  - **Playing area** - The cardboard piece troubled a few players who accidentally moved it during game sessions. Restraining the cardboard was suggested to avoid these issues. Most players felt that the play area was too small. One noted the contrary. The cardboard was used during user tests, because initial testing showed it provided the best backdrop for detecting tags (which was deemed a high priority). The results yielded in Section 11.1.1 suggest that tags would have been almost equally detectable upon a table top surface. Using the table top would open up for a larger and static play area.
  - **Game pieces** - A number of players felt that the weight of the game pieces (to which the tags are affixed) were to light and flimsy. One player felt that they had just the right weight.
  - **More feedback** - Every player commented on the in-game feedback. Most players would have appreciated additional visual feedback (especially in regards to tracks connecting properly) and additional audial feedback.

- **Multiple areas of focus** - Reactions to the multiple areas of focus (playing area and screen) were mixed. Some players appreciated the challenged it presented. One player explicitly denounced the aspect and saw it as a general annoyance. Most players agreed that limiting the areas of focus to one location would simplify the game, but some players were concerned that this might make the game less interesting due to a lack of challenge. An especially interesting comment came from a player, who had played the prototype under less than ideal light conditions, due to cloudy skies. He noted that the lag produced from the extended exposure time of the camera, made it frustrating to focus on the screen. Especially due to his own hands being delayed on-screen.

- **Tag Colors** - When asked if players noticed the difference of colors between the tags, most players responded positively. A few also explained that they used the blue colored tags as anchor points between the placement of tags in-front of them, and how the same situation looked on-screen.

Apart from a few notable exceptions, the results obtained from the interview indicate that players had no trouble using the tactile interface, or figuring out how to affect the game in the way they intended. The notable exceptions include players not being able to see train tracks on obscured tags and a lack of additional feedback. I believe that need for feedback would have been less commented upon, if the train did not fail to transfer on occasion. Resolving this

issue and further testing would prove useful in this case. The problem of players being unable to see train tracks, on obscured tags, can be approached in a number of ways. The easiest, and most often suggested, solution would be to indicate the type of track on the actual tag. However, one of the game mechanics was that the train tracks remained indiscernible until play started. An alternative approach would be to alter the tags physical shape to encourage players not to obscure the tag. Yet another approach would be to use a more robust detection algorithm, allowing the tag to be partially obscured. One such algorithm is briefly discussed in Section 11.4.

I believe that some of the responses from players can be partially attributed to previous lack of experience with the interface. For example, a number of players noted preferring the tactile interface to a mouse or keyboard. However, they might not feel the same way, if they have had to use the interface for an extended period of time, or in a completely different application.

The answers gathered in this section provide valuable insight into the concerns and problems users may experience with this type of interface. I believe it can serve as a foundation upon which a thorough set of user testing could be performed.

11.3 Informal product comparison

In addition to the internal and external testing of the developed prototype, a proprietary commercial product was also tested. The goal was to investigate if it also suffered from some of the issues discovered in the developed prototype. The product in question is the aforementioned Eye Of Judgment [Wikd], released exclusively for the PlayStation 3. Due to the commercial nature of the product, access to the product was restricted to a users perspective. Therefore, the product could only be evaluated externally and conclusions only based on observed behavior. The game setup and an associated playing card is shown in Figure 29.

The testing focused on light sensitivity, detection sensitivity and an overall impression of the pose estimation. The list below details all significant discoveries made while testing the Eye of Judgment:

- **Detection pattern sensitivity** - Imperative testing leads me to believe that the identification pattern on the Eye of Judgment playing cards consist of the black patterns on the top and bottom of the playing card, as well as the four arrows in the middle. Note that "patterns" refers to the shape of the black bars, and not the actual markings inside them. I believe the markings are there for purely aesthetic purposes. The continuous, and thickest, black bar at the top of the card is most likely the guide bar mentioned by Rekimoto and Ayatsuka [RA00], used to detect a potential playing card. Each card has the exact same thick bar at the top, while the smaller black markings vary on each card, and is probably the identification pattern.

  Similar to the prototype developed for this thesis, occluding any of the aforementioned card markings slightly, prohibit the card from being detected. This includes both the various black markers, as well as any of the arrows. In contrast to the prototype, the card offers a fairly large surface, which is not involved in the detection and can be fully occluded without any detrimental affects.

- **Light sensitivity** - Extreme light conditions had a negative impact on the card detection. A brightly and directly lit area caused tags, outside of the light, not to be

Figure 29: **Eye Of Judgment Test setup and playing card.** - Shown left is the setup for the commercial product Eye of Judgment. Shown right is one of the play cards included in the original set.

detected properly. The testing procedure for the developed prototype revealed that in a direct light situation, a number of tags are rendered unrecognizable due to the light being reflected off the tags, and directly into the camera. This issue is neatly avoided in the Eye of Judgment, because the camera views the playing field at an angle and avoids most of the light being reflected. In very low light conditions, camera noise became very visible, and appeared to cause frequent detection problems. Certain cards would continuously switch between being detected and not detected, while remaining static. Photos of the described light scenarios, used with the Eye of Judgment, are included on the supplemental DVD.

- **Delayed reaction** - Regardless of light intensity, the virtual elements seemed to react with a slight delay to changes in front of the camera. It is likely that the development team determined that it was best to let virtual elements transition between places, instead of instantly "warping" in-between positions. It is also possible that the game uses a small portion of time to double check for a cards location and existence, before altering the virtual world to reflect the changes in the real one.

- **Pose Estimation** - Overall, the pose estimation used in the Eye of Judgment appears to be both stable and accurate. Exactly how accurate the pose estimation is, is virtually impossible to determine without deeper access to the games internal code. I find it fitting to note that the pose estimation developed for the prototype also appears accurate when observed during use. However, testing has clearly shown that it is less accurate than the OpenCV counterpart and is also perhaps the cause of the abrupt endings to the game sessions. The time I spent playing the Eye of Judgment did not reveal game mechanics that appeared to require as precise interaction as used in the developed prototype. Therefore, even a black box test of the pose estimation would prove challenging.

The Eye of Judgment is a good example, of how hard it is, to get acceptable results in any given situation. Clearly, the detection used in the game could still use improvement and has flaws similar to that of the developed prototype. I would speculate that quality of the pose estimation has been prioritized to work optimally in select situations. The commercial product would not generate further sales by making it capable of being played in darkness. Therefore, the tracking and pose estimation, in the Eye of Judgment, is in all likelihood intentionally left incapable of handling every situation. It should be noted that, the PlayStation Eye camera [Wikk] used for the game delivers the same resolution and frame-rate as the camera used in this thesis, but it appears to deliver images of lower quality. The playing surface it observes is $30cm \times 30cm$ compared to the $42cm \times 30cm$ used in this thesis.

11.4 Further study and future improvements

This project has personally been my first step into working with augmented reality. It should therefore come as no surprise, that it an immense number of clear-cut possible improvements exist. I intend to describe the ones I find most significant in the following list:

- **Improved Tag filtering** - As described in Section 9.3, an extensive number of contours are searched for the presence of potential tags. How deep to delve into certain contour branches can be determined by examining their size during traversal. If a contour is too small to contain a tag, it should not be further examined. This would improve detection and conserve calculations. The tag filtering process could also be improved by calculating whether the detected square, is a valid projection of a square with equally long sides.

- **Alternate Tag Detection Algorithm** - Apart from increasing the overall robustness of the implemented algorithm, Mark Fiala introduces a technique [Fia04a, Fia04b], which even allows for the tag to be partially occluded and still properly detected. A thorough comparison, and implementation of this new approach, would be a top priority due to the number of comments received from players, regarding tag occlusion.

- **Head mounted displays** - Although a number of players were concerned regarding the potential lack of challenge caused by only needing to focus on one area, implementing the prototype using a head mounted display is ideal for further study. I believe that by using head mounted displays one of the largest barriers between the virtual and real world is removed, which could potentially lead to a better overall experience.

- **Multiple Cameras** - Multiple cameras would improve pose estimation and reduce occlusion caused tag misdetection. However, the added complexity might be too large, to justify the benefits.

- **Pose estimation improvement** - As clearly shown in Section 11.1.2, the pose estimation implemented in this thesis, is less precise than the algorithm provided by the OpenCV library. Hartley and Zisserman [HZ04] show that data normalization significantly improve the calculated homography, and would be an ideal improvement to the implemented algorithm. Other possible improvements include using multiple tags to improve pose estimation and minimize problems related to tag occlusion.

- **Overall prototype refinement** - A number of improvements could be implemented based on the players suggestions, such as visual feedback, audial feedback, more game goals, etc. The prototype also has a number of program related issues which could be addressed, such as properly cleaning up used memory.
- **Alternate applications** - Implementing other games (such as ones mentioned in Section 7), or applying the interface to other tasks, would further test the benefits and problems regarding the interface. A few players suggested using the interface to computer aided design, which I believe would be an ideal choice warranting further study.
- **Improved pose estimation testing** - The testing procedure described in Section 11.1.2 uses averaged values, of all the tags poses and orientations, to estimate a common plane. The procedure could be improved by more accurately measuring the plane the individual tags are supposed to represent. Under the assumption that the largest portion of tags are estimated correctly, the plane could be estimated by iteratively altering its position and orientation until the overall deviation from each tag is smallest. In other words, the most tags would deviate the least from this estimated plane.
- **Thorough testing using refined prototype** - The program and user testing performed in this thesis, has made clear which areas of the prototype could use further refinement. By addressing the issues raised by the individual test protocols, and expanding the number of participants in each test, more conclusive and precise measurements could be achieved.

12 Conclusion

As this is the final section, it is only fitting to draw conclusions, based on the accomplishments in this thesis. The most significant ones are detailed in the following list:

- **Real-Time Tag tracking and pose estimation** - Fully functioning real-time tracking and pose estimation has been described and implemented.
- **Augmented reality board game** - A number of game concept candidates have been considered and described. One has been implemented as a working prototype.
- **Tracking and pose robustness** - The robustness of tracking and estimating of poses has been measured and compared to another similar implementation. An informal comparison with an existing commercial product has also been conducted.
- **User testing** - The developed prototype has been submitted to a qualitative testing process which made use of the "think-aloud" testing protocol.
- **Future improvements** - The testing has shown a large room for improvement, both in regards to the game, the interface and the tracking software. Possible solutions to these issues have been considered and detailed.

Apart from the accomplishments described in the list above, I have personally learned a great deal from the project. I am almost more fascinated by the technology now than I was before. There has only been a single commercial product, in recent years, which really makes use of the technology. In addition to that, it only scratches the surface of the possibilities provided by augmented reality. The prototype game developed and results gathered in this thesis, indicate that the use augmented reality is both viable and usable. Several of the related projects also pave the way for this ideology, and I hope to see more of this technology in connection with games in the future.

Appendices

A Interview Questions

1. How was it?
2. How was the experience of playing with the game pieces?
3. Was it a challenging to move the game pieces around? Did it require more of your concentration, than actually playing the game?
4. Was it difficult to control the game using the game pieces?
5. Was the game logical? What made sense? What did not make sense?
6. Did the game increase in difficulty?
7. What are your thoughts on how to make the game easier?
8. What are your thoughts on the act of coordinating what happened in-front of you, with what was happening on the screen?
9. Would the game be easier if you were not required to observe the screen during play?
10. What made the game difficult?
11. Was it more challenging to ”move the bricks” or ”play the game”?
12. If you could change anything in the game, what would it be?
13. What else do you think this type of interface can be applied to?
14. Did the different colors of the game pieces make a difference during the play session?

References

[Aut] Various Authors. Simple text output. http://www.ogre3d.org/wiki/index.php/Simple_Text_Output.

[Bax] William Baxter. The image debugger. http://billbaxter.com/projects/imdebug/.

[Bou] Jean-Yves Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[BR05] Oliver Brimber and Ramesh Raskar. *Spatial Augmented Reality: Merging Real and Virtual Worlds*. A K Peters, Ltd. , ISBN: 1-56881-230-2, 2005.

[CDS+00] Ben Close, John Donoghue, John Squires, Phillip De Bondi, Michael Morris, Wayne Piekarski, Bruce Thomas, Bruce Thomas, and Unisa Edu Au. Arquake: an outdoor/indoor augmented reality first person application. In *In 4th Int'l Symposium on Wearable Computers*, pages 139–146, 2000.

[Con] Numerous Contributors. Open computer vision library (opencv). http://sourceforge.net/projects/opencvlibrary/.

[CWG+03] Adrian David Cheok, Fong Siew Wan, Kok Hwee Goh, Xubo Yang, Wei Liu, Farzam Farbiz, and Yu Li. Human pacman: A mobile entertainment system with ubiquitous computing and tangible interaction over a wide outdoor area. pages 209–223. Mobile HCI, 2003.

[DB] Joshua Driggs and Bayo. Donkey kong arcade at the quakecon 2005, without the background, scaled and transformed. Creative Commons 2.0, cc-by-sa-2.0. Original Image by Joshua Driggs and edited (Background Removal) by Bayo.

[DBVL+] Jonas Drewsen, Simon Brettschneider, Bue Vendel-Larsen, Daniel Laursen, and Lasse Laursen. Steamwinter. http://steamwinter.com.

[Eri] Sony Ericsson. W810i sony ericsson mobile phone. http://www.sonyericsson.com/cws/products/mobilephones/specifications/w810i.

[Fia] Mark Fiala. Artag. http://www.artag.net/.

[Fia04a] Mark Fiala. Artag, an improved marker system based on artoolkit. *National Research Council Canada*, July 2004.

[Fia04b] Mark Fiala. Artag revision 1. a fiducial marker system using digital techniques. *National Research Council Canada*, November 2004.

[FSH04] Tracy Fullerton, Christopher Swain, and Steven Hoffman. *Game Design Workshop: Designing, Prototyping, and Playtesting Games*. CMP Books, ISBN: 1-57820-222-1, 2004.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0-52154-051-8, second edition, 2004.

[Kat] Hirokazu Kato. Artoolkit. `http://www.hitl.washington.edu/artoolkit/`. Supported by the Human Interface Technology Laboratory (HIT Lab) at the University of Washington, HIT Lab NZ at the University of Canterbury, New Zealand, and ARToolworks, Inc, Seattle.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, Washington, DC, USA, 1999. IEEE Computer Society.

[Ken01] Steven L. Kent. *The Ultimate History of Video Games.* Three Rivers Press, New York, ISBN: 0-7615-3643-4, 2001.

[Köf07] Christina A. Köffel. Heuristics for tabletop games. Master's thesis, Hagenberg Campus of the Upper Austria University of Applied Sciences, September 2007.

[Kni] Gareth Knight. Amiga history guide. `http://www.amigahistory.co.uk/virtuality.html`.

[Kom] Jari Komppa. Extremely simple capture api. `http://sol.gfxile.net/code.html/`.

[Lil03] Björn Liljequist. Planes, homographies and augmented reality. Master's thesis, Lund University, Faculty of Engineering, September 2003.

[Log] Logitech. Logitech quickcam pro 9000. `http://www.logitech.com/index.cfm/webcam_communications/webcams/devices/3056&cl=US,EN`.

[Ltd] A-Rage Pty Ltd. A-rage. `http://www.a-rage.com/`.

[Met] Metacritic.com. Eye of judgment, the (ps3: 2007): Reviews. `http://www.metacritic.com/games/platforms/ps3/eyeofjudgement`.

[Mic] Microsoft. Direct show. `http://msdn.microsoft.com/en-us/library/ms783323(VS.85).aspx`.

[Mic08] Microsoft. Visual studio 2008 professional. `http://msdn.microsoft.com/en-us/vstudio/default.aspx`, 2008.

[Mol94] Rolf Molich. *Brugervenlige Edb-Systemer (in Danish).* Teknisk Forlag, 1994.

[Net] Paul Nettle. Memory manager. `http://www.paulnettle.com/pub/FluidStudios/MemoryManagers/`.

[Oli] Julian Oliver. Levelhead. `http://julianoliver.com/levelhead`.

[Pic80] Peter A. Piccione. In search of the meaning of senet. *Archaeology*, pages 55–58, July/August 1980.

[RA00] Jun Rekimoto and Yuji Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. In *Designing Augmented Reality Environments*, 2000.

[Rek98] Jun Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Asia-Pacific Computer & Human Interaction (Apchi '98)*, 1998.

[SEG98] Zsolt Szalavári, Erik Eckstein, and Michael Gervautz. Collaborative gaming in augmented reality. In *In ACM Symposium on Virtual Reality Software and Technology*, pages 195–204, 1998.

[STC] Loki Software, Creative Technology, and Apple Computer. Open audio library (openal). `http://connect.creativelabs.com/openal/default.aspx`.

[Str] Bjarne Stroustrup. The c++ programming language. `http://www.research.att.com/~bs/C++.html`.

[Sut68] Ivan E. Sutherland. A head-mounted three dimensional display. pages 757–764, 1968.

[SWJ+] Steve Streeting, Justin Walsh, Brian Johnstone, Assaf Raman, and other numerous contributors. Object-oriented graphics rendering engine (ogre3d). `http://www.ogre3d.org/`.

[Sza99] Zsolt Szalavári. *The Personal Interaction Panel - a two-handed Interface for Augmented Reality*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 1999.

[Tru] Trust. Mini hires webcam wb-3300p. `http://www.trust.com/products/product_detail.aspx?item=14776`.

[vgc] vgchartz.com. Eye of judgement (ps3) — vg chartz.com. `http://www.vgchartz.com/games/game.php?id=12350`.

[WAM+] Daniel Wagner, Clemens Arth, Alessandro Mulloni, Tobias Langlotz, and Lukas Gruber. Studierstube tracker. `http://studierstube.icg.tu-graz.ac.at/handheld_ar/stbtracker.php`.

[Wel93] Pierre D. Wellner. Adaptive thresholding for the digitaldesk. Technical Report EPC-1993-110, Rank Xerox Research Centre, Cambridge Laboratory, 61 Regent Street, Cambridge CB2 1AB, 1993.

[Wika] Wikipedia. Carcassonne (board game). `http://en.wikipedia.org/wiki/Carcassonne_(board_game)`.

[Wikb] Wikipedia. Doom: The boardgame. `http://en.wikipedia.org/wiki/Doom:_The_Boardgame`.

[Wikc] Wikipedia. Doom (video game). `http://en.wikipedia.org/wiki/Doom_(video_game)`.

[Wikd] Wikipedia. The eye of judgment.
http://en.wikipedia.org/wiki/The_Eye_of_Judgment.

[Wike] Wikipedia. Jenga.
http://en.wikipedia.org/wiki/Jenga.

[Wikf] Wikipedia. Mahjong.
http://en.wikipedia.org/wiki/Mahjong.

[Wikg] Wikipedia. Mikado (game).
http://en.wikipedia.org/wiki/Mikado_(game).

[Wikh] Wikipedia. Monopoly: The card game.
http://en.wikipedia.org/wiki/Monopoly:_The_Card_Game.

[Wiki] Wikipedia. Pac-man.
http://en.wikipedia.org/wiki/Pac-Man.

[Wikj] Wikipedia. Pipe mania (video game).
http://en.wikipedia.org/wiki/Pipe_Mania_(video_game).

[Wikk] Wikipedia. Playstation eye. http://en.wikipedia.org/wiki/PlayStation_Eye.

[Wikl] Wikipedia. Quake.
http://en.wikipedia.org/wiki/Quake.

[Wikm] Wikipedia. Tower defense.
http://en.wikipedia.org/wiki/Tower_defense.

[Wikn] Wikipedia. Trivial pursuit.
http://en.wikipedia.org/wiki/Trivial_Pursuit.

[Wiko] Wikipedia. Xbox live vision.
http://en.wikipedia.org/wiki/Xbox_Live_Vision.