Eindhoven University of Technology
Department of Mathematics and Computer Science
Visualization Group

MASTER THESIS

AUGMENTED REALITY FOR INDOOR APPLICATIONS ON MOBILE DEVICES

T.K.A. Crijns

February 7, 2012

University	Eindhoven University of Technology (TU/e) Department of Mathematics & Computer Science Den Dolech 2 5612 AZ, Eindhoven 040-2479111
Company	Alten Beukenlaan 44 5651 CD, Eindhoven 040-2563080
Supervisors	Dr. M.A. Westenberg (TU/e) Ir. D.W. Hardy (Alten) Drs. A.A.G. Willems (Alten)
Date & Place	February 7, 2012 Eindhoven

PREFACE

This master thesis is the result of my graduation project which concludes my Computer Science and Engineering study at the Eindhoven University of Technology. The project was assigned by the Visualization Group of the TU/e and executed externally at Alten PTS. The project started in September 2011 and was set to take 6 month in order to complete.

I would like to thank my supervisors Ir. D.W. (David) Hardy and Drs. A.A.G. (Ard) Willems for their support and giving me the opportunity to perform my graduation project at Alten. I would also like to thank my supervisor at the TU/e Dr. M.A. (Michel) Westenberg for his guidance and active involvement during the course of this project. Lastly I would like to thank Dr. C. (Kees) Huizing for taking place in the exam committee.

ABSTRACT

The research topic of this thesis is augmented reality (AR). It is a technique which enables the user to see computer-generated objects inside a real environment; this is often accomplished by using displays in order to augment parts of the scenery with virtual objects. Augmented reality is a wide-ranging research field and has many possible applications like in architectural designs, medical examinations and navigation.

Mobile devices such as smartphones and tablets are becoming more powerful these days w.r.t. their processing speed and memory. Because of their compact design they provide an excellent platform for augmented reality (e.g. by capturing images from the camera and using them to create an onscreen augmented world).

Real-time augmented reality relies on knowing the user's location and orientation. In order to extract this information different methods exist. A straightforward idea is to simply use GPS; unfortunately GPS has its limitations, for example it cannot be used indoors and doesn't provide information on the user's orientation. To overcome this drawback it can be used alongside orientation sensors (e.g. accelerometers, magnetometers and gyroscopes). Another approach is to use computer vision techniques, however, in general they are computationally intensive.

In this thesis we propose a method that combines orientation sensors with computer vision in order to determine the location and orientation of the device. It is used to determine the location of the virtual object. The orientation is calculated by combining accelerometers, magnetometers and gyroscopes. By using the orientation of the device we accelerate the computer vision computations without losing accuracy. The processing time for the orientation and computer vision routines require 35%. On the other hand, drawing and frame processing consumes 65%, with a frame rate of 1.9 fps. It is a consequence of the expensive camera frame conversions that are implemented in software. For future work the focus should be on implementing the camera functionality in hardware.

CONTENTS

1 Introduction

Augmented reality or AR for short is a technique which enables the user to perceive in real-time computer-generated objects inside a real environment. This is often accomplished by using displays in order to augment parts of the scenery with virtual objects. Different types of displays can be used, for example; head mounted displays, LCD displays on phones or spatial projection into the real environment. This graduation project will research the possibilities of using AR to visualize virtual objects on mobile phones. The scenery is augmented with 3D models which are displayed on a screen to the user. Augmentation of the scenery is accomplished by using the different available hardware components; like GPS, movement sensors, orientation sensors and camera.

1.1 Motivation

AR has many interesting and useful applications. For instance it can benefit the development and (re)construction of buildings, as shown in figure 1. Instead of looking at rendered scenery users can walk around and see the building as if it was actually there by projecting it in real-time on a screen. In this particular example a large marker was placed, it functions as a fixed-point for the system and is used to extract the location and orientation of the virtual building. The Brazilian developers of Rossi Residential hold the Guinness book of world records for producing the largest augmented reality experience (10,000 Sq. Ft. Augmented Reality Marker Sets Guinness World Record).

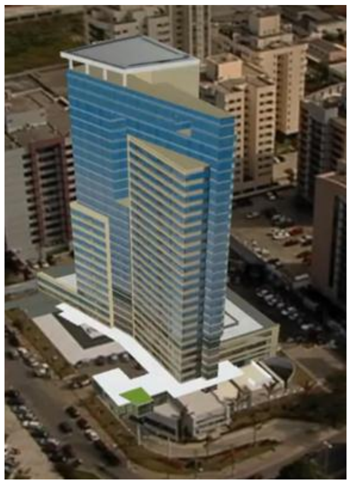


Figure 1 The building is projected onto the original image to give an impression of how the building will be, once completed (10,000 Sq. Ft. Augmented Reality Marker Sets Guinness World Record).

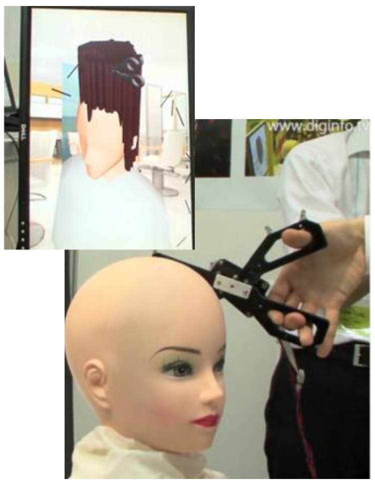


Figure 2 Virtual hair is augmented with the real scenery to train hairdressers (Air Hair: Augmented reality for those who want to learn haircutting).

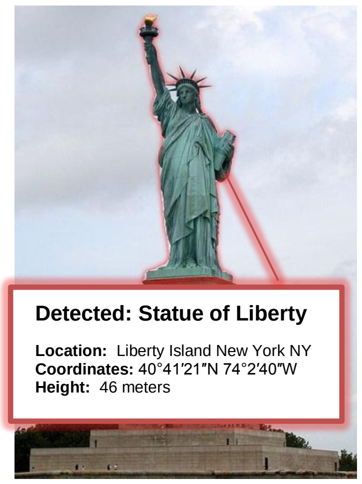

Detected: Statue of Liberty
Location: Liberty Island New York NY
Coordinates: 40°41′21″N 74°2′40″W
Height: 46 meters

Figure 3 AR used to provide information about the currently visible content (Elcobbola, 2010).

In addition to construction purposes AR could be used to let people learn and improve certain skills. In figure 2 we can see a physical head that is projected on a screen augmented with virtual hair. It was used in Tokyo to let student hairdressers practice and improve in their field of expertise (Air Hair: Augmented reality for those who want to learn haircutting).

The application in figure 3 is an entirely different approach of AR, instead of projecting virtual 3D objects extra information is displayed inside the image. This data is relevant to the content that is displayed at that moment; it provides the user with information about the scenery. In the above example the statue of liberty has an overlay which provides additional information, this is accomplished by detecting objects inside the scenery and retrieving related information.

There are many other useful applications; like in games or the projection of advertisements during football matches (virtual boards with ads are displayed adjacent to the goals). The specific application we focus on has most in common with the example shown in figure 1. In our approach we want to create augmented scenery based on the position of the user like in figure 1 however we don't want to depend on the usage of markers. Markers have the major advantage that it is less computational intensive than non-marker approaches, however the drawback of this approach is that it makes assumptions on the environment and limits the usability. Another dissimilarity with the method shown in figure 1 is that we want to deploy it in indoor environments. Alten suggested the usage of visualizing indoor object, like pipes inside a room shown in figure 4. Accordingly, the project was performed at Alten and developed with these intentions in mind.

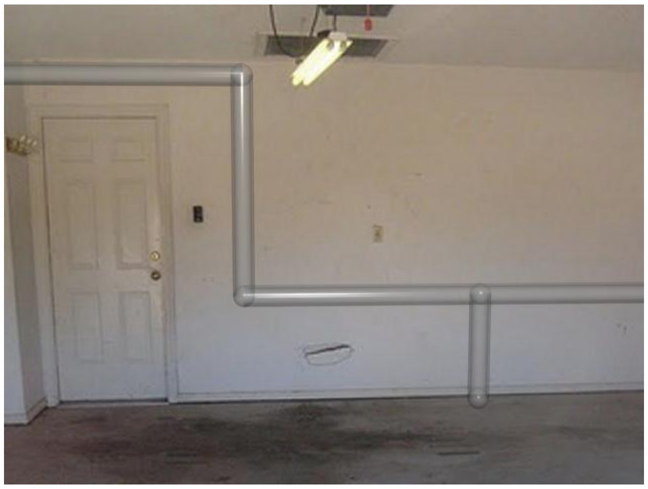


Figure 4 The room is augmented with pipes to aid in construction work (Winn, 2009)

1.2 OUTLINE

The remainder of this thesis is outlined as follows; in the following section we will provide the different applications that could benefit from the usage of augmented reality. In chapter 2 we will give a problem definition and clarify some basic concepts that should be known before we go into the topic of creating AR. The final part of this chapter is dedicated to the functional and non-functional requirements that should be adhered to. In chapter 3 we focus on related work that has been done, this includes the research performed on a similar topic by a previous student David Hardy (2011). In section 3.1 we will study recent developments and see how they are currently used. The different methods and techniques to created augmented reality are discussed in the paper of Crijns (2011) that was created in preparation of this thesis. The solutions to problems shown in chapter 2 are presented in chapter 4 "Approach"; the actual implementation of this is discussed in the subsequent chapter. As a final point we will conclude with discussing the results and indicate possible improvements of the implementation.

2 Problem definition

In this chapter we define the problem in creating augmented reality applications. We will start with a general and intuitive description and continue with providing an introduction to viewpoints and matrices. These concepts are needed in order to describe transformations which are used in the formal problem definition (section 2.4). We will conclude this chapter with the requirements that should be satisfied.

2.1 General description

There are many useful applications for augmented reality, however if we want to adequately create the illusion of AR on mobile devices the location of the user/device should be known.

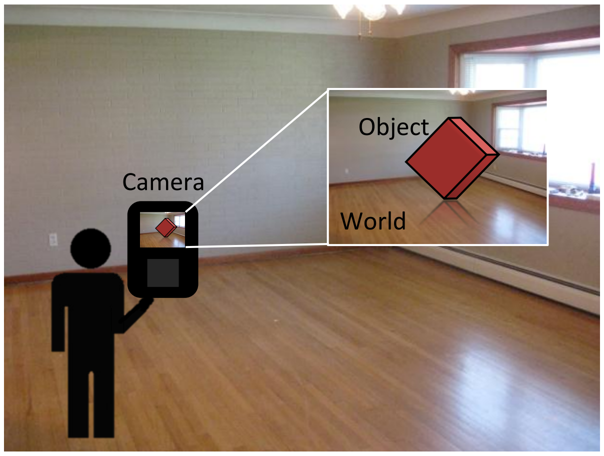


Figure 5 Inside this scene a cube is projected as an augmented object (Duce, 2010).

Let's have a look at the example in figure 5, in order to correctly project the cube inside the scene we need to know the location and orientation of the cube (or in general the object) in relation to the world; furthermore we need to know the location and orientation of the camera in relation to the world. The location and orientation of the cube is predefined, what remains is reconstructing the location and orientation of the camera. A location in 3D space is simply a transformation; orientation and translations can be described using transformations that are presented in chapter 2.3. In order to formally describe relative transformations we first need to define coordinate systems.

2.2 Coordinate systems

A single point or location in a right handed 3D Cartesian coordinate system can be represented in different ways. We distinguish three coordinate systems; Camera C, World W and Object O. In the three examples below we have three points, each of these points are described in relation to a different coordinate system.

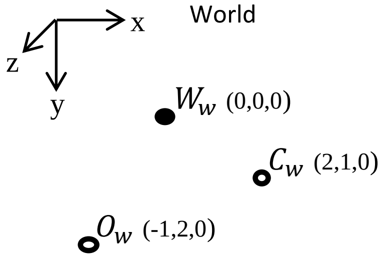


Figure 6 Points represented in a world-oriented coordinate system.

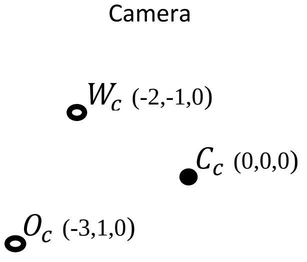


Figure 7 Points represented in a camera-oriented coordinate system.

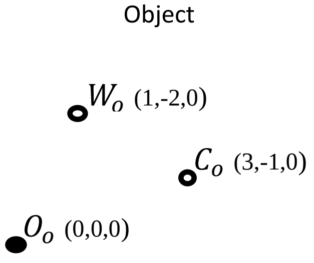


Figure 8 Points represented in an object-oriented coordinate system.

In the above example we can see that it is possible to describe the same set of points from different coordinate systems. If the location of a point P is represented w.r.t. the world we write P_w, in a similar way we write P_c and P_o.

2.3 TRANSFORMATIONS

Points are transformed from one coordinate system to the other by applying a rotation (shown in eq. 2) and a consecutive translation (shown in eq. 1) which can be represented by transformation matrices. We define 0_k as the zero-matrix with k rows and one column, furthermore I_n is defined as the identity matrix with n rows and n columns. A translation of point $P = [x, y, z, 1]^T$ by t_x, t_y and t_z can be described by a translation matrix t.

$$\begin{bmatrix} & t_x \\ I_3 & t_y \\ & t_z \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \qquad \text{Eq. 1}$$

$$\equiv tP = P'$$

Similarly a general rotation of point $P = [x, y, z, 1]^T$ with ψ, θ and φ can be described by means of a rotation matrix R.

$$\begin{bmatrix} 1 & 0 & 0 & \\ 0 & \cos\psi & -\sin\psi & 0_3 \\ 0 & \sin\psi & \cos\psi & \\ & 0_3^T & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta & \\ 0 & 1 & 0 & 0_3 \\ -\sin\theta & 0 & \cos\theta & \\ & 0_3^T & & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & \\ \sin\varphi & \cos\varphi & 0 & 0_3 \\ 0 & 0 & 1 & \\ & 0_3^T & & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \qquad \text{Eq. 2}$$

$$\equiv RP = P'$$

Where ψ defines the rotation in radians around the x-axis, θ defines the rotation in radians around the y-axis and φ represents the rotation in radians around the z-axis. By putting these rotation and translation matrices together we can create a transformation matrix Δ.

$$\Delta = \begin{bmatrix} & t_x \\ I_3 & t_y \\ & t_z \\ 0_3^T & 1 \end{bmatrix} R = \begin{bmatrix} c_\theta c_\varphi & -c_\theta s_\varphi & s_\theta & t_x \\ c_\psi s_\varphi + c_\varphi s_\theta s_\psi & c_\varphi c_\psi - s_\theta s_\varphi s_\psi & -c_\theta s_\psi & t_y \\ -c_\varphi c_\psi s_\theta + s_\varphi s_\psi & c_\psi s_\theta s_\varphi + c_\varphi s_\psi & c_\theta c_\psi & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. 3}$$

With $\cos\psi = c_\psi, \sin\psi = s_\psi, \cos\varphi = c_\varphi, \sin\varphi = s_\varphi, \cos\theta = c_\theta, \sin\theta = s_\theta$

If we want to go from a world to camera coordinate system we need to apply the transformation matrix Δ_w^c to point P_w, thus $P_c = \Delta_w^c P_w$. It is also possible to transform back using the inverse matrix.

$$\Delta_a^b = (\Delta_b^a)^{-1} \qquad \text{Eq. 4}$$

The inverse matrix transformation can be calculated using the following identities; for matrix multiplication it holds that $B^{-1}A^{-1} = (AB)^{-1}$ and because our rotation matrix is orthogonal (Bellman, 1997, p. 15) (Watkins, 2002, p. 192) we have that $R^{-1} = R^T$ (Watkins, 2002, p. 187). Finally the inverse of a translation matrix $T^{-1} = -T$ now we can derive;

$$\Delta^{-1} \qquad \text{Eq. 5}$$

$$= \left(\begin{bmatrix} R & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ 0_3^T & 1 \end{bmatrix}\right)^{-1} = \left(\begin{bmatrix} I_3 & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ 0_3^T & 1 \end{bmatrix}\begin{bmatrix} R & 0_3 \\ 0_3^T & 1 \end{bmatrix}\right)^{-1} = \begin{bmatrix} R & 0_3 \\ 0_3^T & 1 \end{bmatrix}^{-1}\begin{bmatrix} I_3 & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ 0_3^T & 1 \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} R^T & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0_3^T & 1 \end{bmatrix}\begin{bmatrix} I_3 & \begin{matrix} -t_x \\ -t_y \\ -t_z \end{matrix} \\ 0_3^T & 1 \end{bmatrix} = \begin{bmatrix} R^T & \begin{matrix} -t_x \\ -t_y \\ -t_z \end{matrix} \\ 0_3^T & 1 \end{bmatrix}$$

For sequential composition we can use the formula in eq. 6 (Simon, Fitzgibbon, & Zisserman, 2000, p. 4).

$$\Delta_a^c = \Delta_b^c \, \Delta_a^b \qquad \text{Eq. 6}$$

2.4 Formal description

Now all the preliminaries are defined we can give a formal description of the problem. In the first section of this chapter we mentioned that if the location and orientation of the model is known then we only need to derive the location of the user w.r.t. the world. By using the above equations we can indeed show that if the transformation from the world to the object is known Δ_w^o then we only need to get the transformation of the camera in relation to the world Δ_c^w in order to compute the location of the object from a camera coordinate system Δ_o^c.

$$\begin{aligned} &\Delta_o^c \\ &\overset{\text{eq. 6}}{=} \Delta_w^c \, \Delta_o^w \\ &\overset{\text{eq. 4, eq. 5}}{=} (\Delta_c^w)^{-1} (\Delta_w^o)^{-1} \end{aligned} \qquad \text{Eq. 7}$$

We can also show that it is equally sufficient to know the location of the camera in relation to the object Δ_c^o if we want to know the transformation from world to camera space Δ_w^c or vice versa.

$$\begin{aligned} &\Delta_w^c \\ &\overset{\text{eq. 4, eq. 5}}{=} (\Delta_c^w)^{-1} \\ &\overset{\text{eq. 6}}{=} (\Delta_o^w \, \Delta_c^o)^{-1} \\ &\overset{\text{eq. 4, eq. 5}}{=} ((\Delta_w^o)^{-1} \, \Delta_c^o)^{-1} \end{aligned} \qquad \text{Eq. 8}$$

So if we know one of these transformations Δ_c^w, Δ_c^o (or using eq. 4 their inverse Δ_w^c, Δ_o^c) then we can project any object point into the camera coordinate system. How to find such a transformation is presented in chapter 4 “Approach”.

2.5 Requirements

The implementation and approach we take should meet specified requirements, which can be split into functional and non-functional requirements. The difference between these requirements is that the functional requirements describe what the system should do in contrast to how it should behave. Non-functional requirements don’t change the behavior or function of the product but describe properties that the system should have (e.g. if we look at performance we can say that the system should run at 30 fps). A detailed list of all the desired requirements can be found in Appendix A.

For the functional requirements we distinguish the following categories; camera, feature-tracking, 3D objects, orientation and peripherals. The camera-related requirements specify what the camera should be able to display and that it is calibrated. Feature tracking requirements indicate what should be tracked. All the functional requirements in relation to the model that should be augmented with the scene are in the 3D objects category. Restrictions on the orientation calculation like calculation of pitch, roll and yaw are listed under orientation. Finally we have the peripherals that specify the components that should be present on the mobile device.

The non-functional requirements are split into performance and usage-related specifications. For performance related issues we have requirements like characterizing system stability and AR immersion. Requirements on how the system should be used are under the last category “usage”.

3 RELATED WORK

In this chapter we will focus on work done in the past and look at some recent developments in the field of augmented reality which include usage in road trafficking and medical surgery. We continue with discussing work that has been done in preparation for and in relation to this thesis. In the last section we summarize the related work and explain how it contributed to our final implementation.

3.1 Recent developments

In this chapter we will have a look at recent developments in different areas. All papers are published from 2011 up to 2012 and cover augmented reality in: vehicles, mobile vision, games and surgery.

3.1.1 AUGMENTED REALITY IN VEHICLE SYSTEM

The paper of (Moussa, Radwan, & Hussain, 2011) studied a newly developed augmented reality vehicle systems, called ARV. The paper had two goals, one to test the performance of the ARV system and the other to test a left-turn maneuver on a two-way stop-controlled intersection. Their findings on a left-turn maneuver are of less interest and will be skipped. In figure 9 we can see the ARV system built into a car.

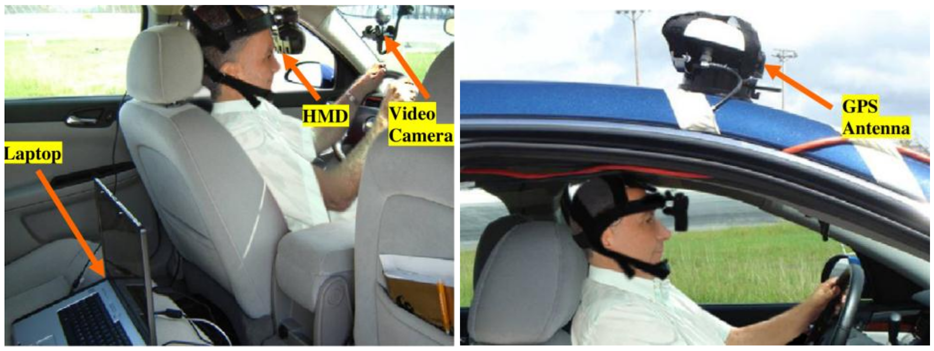


Figure 9 The ARV consisting of a laptop HMD, camera and GPS (Moussa, Radwan, & Hussain, 2011, p. 4).

The laptop was the main component that performed all the processing work. It connected with the GPS module in order to get the location and direction of the vehicle. The camera was needed to acquire the orientation of the vehicle by using image registration techniques. A canny edge detector was used to detect the side lines of the road and together with the intrinsic camera parameters (which are also discussed in chapter 4.2.3) the orientation could be reconstructed. Based on the orientation and location of the driver virtual objects could be rendered inside the camera image which would be displayed to the user via the head mounted display (HMD).

The validation of the ARV system was done by testing the augmented view which had to be free of any visual inconsistency due to scale, orientation, and/or position errors in aligning visual objects with real objects in the final view. It was tested by two test groups, one with the ARV system and the other without. Two scenarios where performed; in one the subjects were asked to drive 40 km/h and as close to the center of the lane as possible (along curved and straight road segments). In the second scenario the driver had to stop the vehicle as close to a stopping line as possible. For both scenarios the distance to the stop line, the average cruising speed and the average offset from the center of the lane where recorded. Student t-tests (Strijbosch, 2006) where applied on the group with the ARV system versus without the ARV system. At a 95% confidence level no significant differences were found during both scenarios when driving along a straight segment, however for the curved segment significant differences were found. The authors attributed the HMD and fixed video camera which slowed drivers down when going along curved roads (Moussa, Radwan, & Hussain, 2011, p. 12). For straight roads the test showed that the system performed adequately without any visual inconsistency.

3.1.2 AUGMENTED REALITY IN MOBILE VISION

The paper of (Hua, Yun, Turk, Pollefeys, & Zhang, 6 December 2011) points out the different challenges and explores recent developments in augmented reality on mobile devices.

There are several important issues that arise when working with mobile devices. First of all we have limited computational capacity which is yet insufficient to handle the large amount of processing required for most augmented reality applications. Secondly a method should be found in order to handle and acquire the vast amounts of data (which is a prerequisite for most mobile vision applications). Finally the paper denotes the issue of extracting contextual information.

Recent developments in the industry includes the SnapTell (2006), Nokia Point & Find (2011) and Google (2012) applications. They all work by matching images with a large database in order to extract useful information. The image is taken by the user from his camera and used to match it with a large pre-annotated database, the found match is retrieved and the annotated information is displayed to the user (see also figure 3).

They conclude that more work on these fields will benefit and improve performance and that it will increase the shift from personal computer to mobile devices. Furthermore they indicate that it is not only about making algorithms faster in order to work in real-time environments, but that it is also about the dynamic integration of information (for example dynamically interfacing with different types of sensors).

3.1.3 AUGMENTED REALITY IN GAMES

We can distinguish different kinds of games (Bonsignore, et al., 2012). For example we have the context aware, alternate reality, social networks and augmented reality games. Augmented reality games can be used to entertain, inspire, motivate or educate people. In the paper of (Yamabe & Nakajima, 2012) a variety of augmented reality games were designed and tested, one of these was a poker game shown in figure 10.

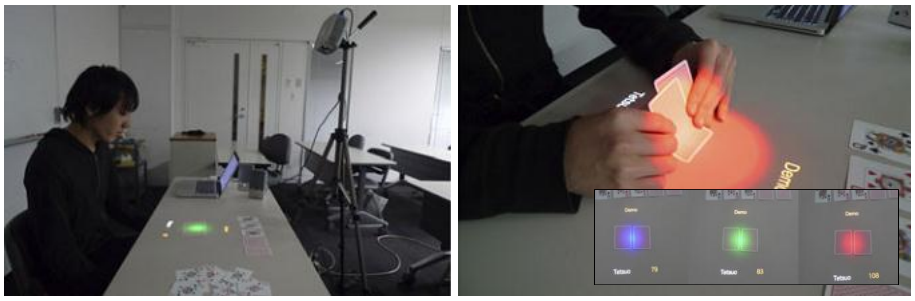


Figure 10 EmoPoker; an augmented reality poker game (Yamabe & Nakajima, 2012, p. 12).

The EmoPoker system is an augmented reality system created to asses emotional state of players during poker games. It can help control the emotional state of a player, e.g. a poker face. We know from studies that emotional excitement can increase the amount of irrational decision-making (Schwarz, 2000, pp. 433–440) and decrease cognitive skills. EmoPoker assesses the players emotional state by using a heart rate monitor. In order to track the poker cards RFID readers were embedded inside the table and RFID tags were placed inside the cards. Based on this information a beam of light is projected onto the cards inside the players hand, the light color indicated the excitement level of the player. Blue means calm and red excitement which is also shown in figure 10. Another usage of this system is to balance poker skills between players in a group. This can be accomplished by hiding the excitement level for novice players and using earphones to indicate heart rate for self-assessment. Tests were performed in order to examine the usefulness of earphones. Their conclusion was that there was no statistical difference (e.g. p < 5%) between using and not using earphones (Yamabe & Nakajima, 2012, p. 19).

3.1.4 AUGMENTED REALITY IN ENDOSCOPIC SURGERY

The paper of (Nakamoto, Ukimura, Faber, & Gill, March 2012) studied a system to create augmented reality in endoscopic surgery. The goal was to help surgeons and improve surgery by making it more precise and safe. An example of the system is shown in figure 11.

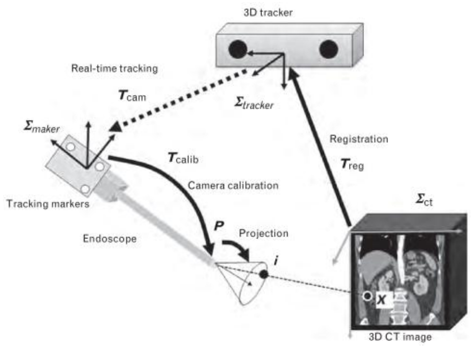


Figure 11 Example of the endoscopic surgery system setup (Nakamoto, Ukimura, Faber, & Gill, March 2012, p. 123).

The three main components are the endoscope, tracker and image display. The tracker determines the camera position and orientation w.r.t. the tracker by using the markers on the endoscope. Subsequently the camera input from the endoscope is augmented and displayed to the specialist. In their application it was used to assists surgeons in achieving a negative surgical margin and maximize the tissue preservation around a tumor.

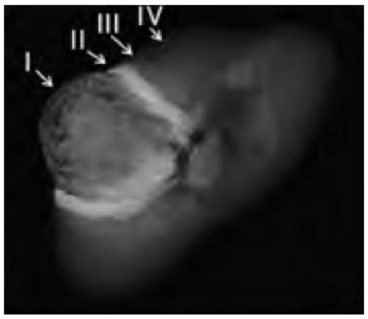


Figure 12 An example of a tumor with corresponding zones (Nakamoto, Ukimura, Faber, & Gill, March 2012, p. 124).

From a CT scan of a tumor color coded zones where created and augmented with the real image from the endoscope. Zone 1 was the tumor itself, zone 2 was a margin of 0 to 5mm around it, zone 3 was 5 to 10 mm and larger than 10 mm was zone 4 (see figure 12).

Their conclusion of the usability of such techniques in order to help physicians was that it created significant advancements in the precision of endoscopic surgery. However there is a major drawback in this approach, it could not track and react to the deformability and dynamic motion of organic tissue which limited the usability.

3.2 PREVIOUS WORK

This thesis is a continuation of the work previously performed by Hardy (2011), the relation to his thesis will be discussed in the following chapter. In preparation of this thesis research was performed on the topic of indoor positioning systems (Crijns, 2011). In this paper the different methods available for indoor tracking were examined, which will be presented in section 3.2.2.

3.2.1 AUGMENTED REALITY FOR LANDSCAPE DEVELOPMENT ON MOBILE DEVICES

The work by David Hardy was performed externally at Alten. It had a similar topic in the sense that it researched augmented reality on mobile devices, however its usage was not for indoor environments; it was designed to work outdoors to support in landscape development (e.g. visualizing buildings or other planned constructions). In figure 14 we can see an example of how it was used.

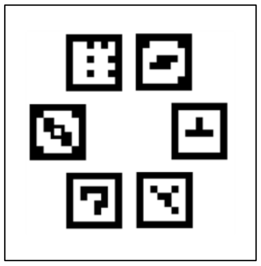


Figure 13 The six different markers that could be recognized (Hardy, 2011, p. 20).

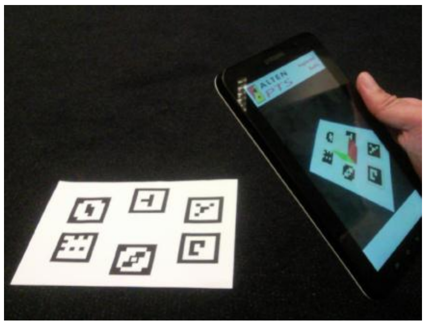


Figure 14 Example of the implementation (Hardy, 2011, p. 50).

A smartphone was used that recognized the different predefined markers and then projected a model (in this example a cube) on the surface. In total there were six types of markers arranged in a circular pattern as shown in figure 13. The projection that Hardy had implemented was accurate, however the drawback was that the method relied on symbols which had to be planar. This thesis is a continuation of that work in the sense that it creates augmented reality and was also developed for mobile devices. Furthermore it removed the dependency of predefined marker usage.

3.2.2 *INDOOR POSITIONING SYSTEMS*

The opportunity of performing the research on indoor positioning systems was provided by dr.ir. H.M.M. (Huub) van de Wetering who is working at the TU/e, it was executed in preparation of this thesis. The paper of Crijns (2011) explains the different methods used to determine the location in indoor environments.

The first technique discussed was the usage of radio identification to determine the position of the user. An example is the placement of RFID chips in a grid like pattern and a scanner that actively scans the chips in order to determine their position.

Another approach is to measure the angle of incoming radio signals using smart antennas, these antennas differ from normal ones in the sense that they can measure the angle of incoming radio waves. The requirement for this method to work is that the origin of the waves should be known.

Measuring the strength of radio signals is another method to determine the position of a device. A convenient implementation of this strategy is using widely available Wi-Fi stations together with trilateration to determine position.

It is also possible to measure elapsed time instead of signal intensity (or strength) like in the previous example. An implementation of this could be speakers and microphone to measure the elapsed time between the departure and arrival time of the sound.

Inertial measurements units (or IMU's) can be used to measure orientation if we would combine it with a GPS receiver than the position could also be determined.

The last discussed method relies on camera vision in order to get the position. It is a costly approach but when done right can be very accurate (Braunegg, 1989, p. 3) (Crijns, 2011, p. 7).

In the table 1 we can see a comparison between the different techniques together with the desired properties. These properties are deployability, accuracy, self-containedness and low setup costs. A system is self-contained if it does not require any external components in order to work. This implies the last property, e.g. directly deployable. The other way around is not necessarily valid. A system that can be used directly is not always self-contained and could rely on external components in order to function.

	Accuracy	Self-contained	Low setup costs	Directly deployable
RFID Radio identification	0/+	-	0/+	-
Smart antennas Radio angle of arrival	+	-	o	-
Wi-Fi **Radio signal strength**	o	-	o	-
Microphones **Sound time of arrival**	o	-	o	-
IMU Inertial measurements	-	+	+	+
Camera Vision	o	+	o	-/0

+ method strongly contributes to this property.

o method partially contributes to this property.

- method weakly contributes to this property.

Table 1 A comparison of the different methods for indoor positioning system (Crijns, 2011, p. 9)

The solution presented in this thesis is a combination of the last two strategies, thus IMU's in combination with vision is used to acquire position.

3.2.3 Markerless approaches

Before the final approach was chosen different papers with markless tracking approaches were studied which include the paper of Simon et al. (2000) and Klein & Murray (2007). In figure 15 we can see the algorithm of Klein & Murray (2007) at work. It shows a pre-initialized map with detected feature points. This map is tracked across the screen and can be used to project virtual objects on.

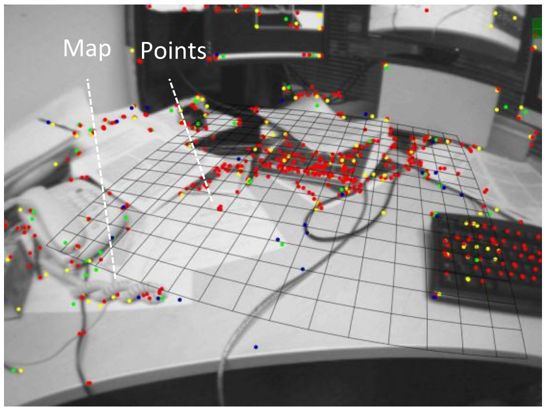


Figure 15 The map extended with feature points (Klein & Murray, 2007, p. 1).

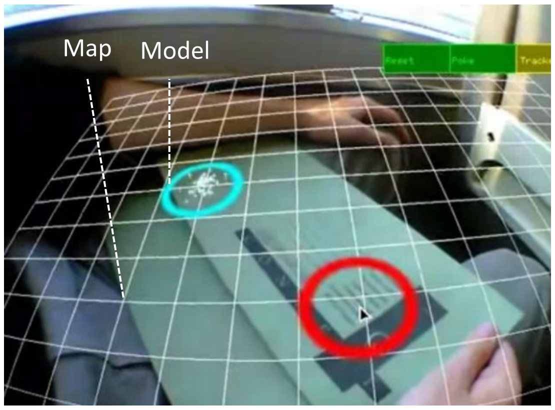


Figure 16 The map is used in order to project virtual object in the scene.

The system is implemented with two processes that run in parallel, one tracks feature points and the other updates and improves the map with new feature points. The tracking process tries to find the orientation of the camera w.r.t. the map by tracking and projecting feature points on the map, the process continuously receives images from the camera and for each image it performs the following steps.

First it estimates the camera orientation based on the detected motion and projects points of the map onto the image using the estimated orientation. The estimated orientation is used by projecting 50 feature points onto the camera image and using the projection error to further improve the orientation. Finally the last step is again performed but this time for 1000 points. The resulting orientation estimation is used to transform world points to camera points. In parallel to this thread we have the mapping process, its task is to improve and extend the map. The map should be constructed in advance and is then used by this process to further improve and extend the map with new feature points. It could for example be constructed using a stereo camera which are two cameras placed in parallel to each other and can be used to extract the location of feature points.

The results of this paper are promising, however it was implemented on a Intel Core 2 Duo 2.66 GHz processor running Linux (Klein & Murray, 2007, p. 6). At least for now these performance requirements did seem too high for the use on mobile devices, it also required a large point set and a pre-computed map in order to work.

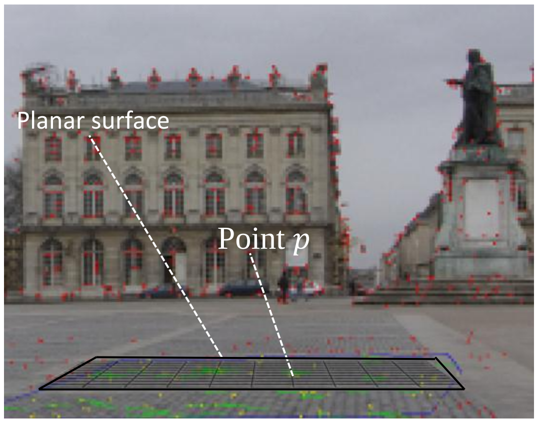


Figure 17 Planar surface with points (Simon, Fitzgibbon, & Zisserman, 2000, p. 5).

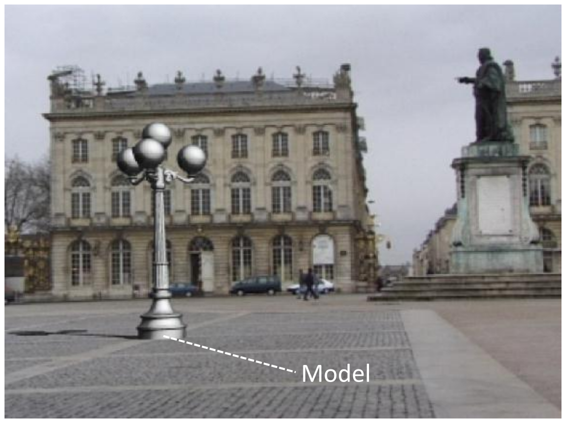


Figure 18 The projected model onto the planar surface (Simon, Fitzgibbon, & Zisserman, 2000, p. 6).

The paper of Simon et al. (2000) used planar homographies in order to project virtual objects. In figure 17 we can see a planar surface and model projected onto this plane. On this plane we can project a model as shown in figure 18. The algorithm uses a world coordinate space wherein the model location is stored, this world coordinate system is projected to the first frame. For each subsequent frame a projection from the previous to the current frame is made. In this way we can project anything from the world into the current frame. Projections from one coordinate system to the others are performed by using planar homographic projections (which are mappings from one 2D surface to another). In figure 19 we can see the different coordinate systems, point p_w (which could be the location of the model) is projected onto camera frame i which is then projected to the subsequent frame $i+1$.

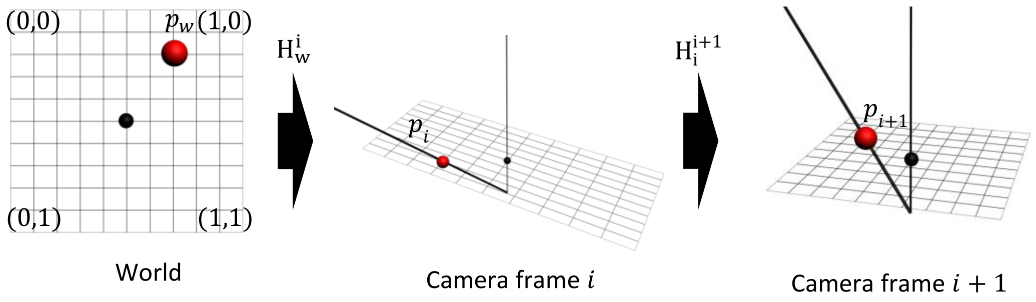


Figure 19 Homographic projections from world to subsequent camera frames.

Mappings are performed using homographic projections denoted by $\mathrm{H}_{\alpha}^{\beta}$ which is a 3 by 3 matrix that defines the projection from any coordinate system α to β. By applying successive mappings we can project a point from a world coordinate system to any camera frame. First we project from the world to the first camera frame $\mathrm{H}_{\mathrm{w}}^{0}$. We can now project to any subsequent camera frame i using $\mathrm{H}_{0}^{\mathrm{i}} = \prod_{k=1}^{i} \mathrm{H}_{\mathrm{k-1}}^{\mathrm{k}}$. Thus we project from the world to camera frame i via $\mathrm{H}_{\mathrm{w}}^{\mathrm{i}} = \mathrm{H}_{0}^{\mathrm{i}}\mathrm{H}_{\mathrm{w}}^{0}$. Given a set of points in the source and destination coordinate space we should be able to calculate a homographic transformation that maps all these points correctly. It should be noted that the homographic projections we are looking for has a scaling factor λ.

$$p_{\beta} = \lambda \mathrm{H}_{\alpha}^{\beta} p_{\alpha} \qquad \text{Eq. 9}$$

The points are equal up to the scaling factor λ. It introduces a set of homogenous equal points, thus for a given z there is a z' such that eq. 10 holds.

$$\begin{bmatrix} x \cdot z \\ y \cdot z \\ z \end{bmatrix} = \mathrm{H}_{\alpha}^{\beta} \begin{bmatrix} x \cdot z' \\ y \cdot z' \\ z' \end{bmatrix} \qquad \text{Eq. 10}$$

By using the fact that we have homogenous equal points we can prove that these points $p_{\beta} = (x \cdot z, y \cdot z, z)$ and $p_{\alpha} = (x \cdot z', y \cdot z', z')$ are in parallel.

$$\mathrm{p}_{\beta} \times \mathrm{H}_{\alpha}^{\beta} \mathrm{p}_{\alpha} \qquad \text{Eq. 11}$$

≡ By definition of p_{α} and p_{β}

$$\begin{bmatrix} \mathrm{x \cdot z} \\ \mathrm{y \cdot z} \\ \mathrm{z} \end{bmatrix} \times \mathrm{H}_{\alpha}^{\beta} \begin{bmatrix} \mathrm{x \cdot z'} \\ \mathrm{y \cdot z'} \\ \mathrm{z'} \end{bmatrix} \qquad \text{Eq. 12}$$

≡ By definition of the cross product

$$\begin{bmatrix} \mathrm{y \cdot z \cdot z' - y \cdot z' \cdot z} \\ \mathrm{x \cdot z' \cdot z - x \cdot z \cdot z'} \\ \mathrm{-x \cdot z' \cdot y \cdot z + x \cdot z \cdot y \cdot z'} \end{bmatrix} \qquad \text{Eq. 13}$$

≡ By simplifying the equation

$$\mathbf{0} \qquad \text{Eq. 14}$$

Hence the of cross product of the homographic projection of two points should be equal to eq. 14. Given a set of points $(p_{\alpha_1}, p_{\alpha_2}, p_{\alpha_3}, p_{\alpha_4})$ in a source coordinate system α and corresponding points $(p_{\beta_1}, p_{\beta_2}, p_{\beta_3}, p_{\beta_4})$ in the destination coordinate system β and solving eq. 11 we derive:

$$\begin{bmatrix} 0 & 0 & 0 & -z_1'x_1 & -z_1'y_1 & -z_1'z_1 & y_1'z_1 & y_1'y_1 & y_1'z_1 \\ z_1'x_1 & z_1'y_1 & z_1'z_1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'z_1 \\ -y_1'x_1 & -y_1'y_1 & -y_1'z_1 & x_1'x_1 & x_1'y_1 & x_1'z_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -z_2'x_2 & -z_2'y_2 & -z_2'z_2 & y_2'z_2 & y_2'y_2 & y_2'z_2 \\ z_2'x_2 & z_2'y_2 & z_2'z_2 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2'z_2 \\ -y_2'x_2 & -y_2'y_2 & -y_2'z_2 & x_2'x_2 & x_2'y_2 & x_2'z_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -z_3'x_3 & -z_3'y_3 & -z_3'z_3 & y_3'z_3 & y_3'y_3 & y_3'z_3 \\ z_3'x_3 & z_3'y_3 & z_3'z_3 & 0 & 0 & 0 & -x_3'x_3 & -x_3'y_3 & -x_3'z_3 \\ -y_3'x_3 & -y_3'y_3 & -y_3'z_3 & x_3'x_3 & x_3'y_3 & x_3'z_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -z_4'x_4 & -z_4'y_4 & -z_4'z_4 & y_4'z_4 & y_4'y_4 & y_4'z_4 \\ z_4'x_4 & z_4'y_4 & z_4'z_4 & 0 & 0 & 0 & -x_4'x_4 & -x_4'y_4 & -x_4'z_4 \\ -y_4'x_4 & -y_4'y_4 & -y_4'z_4 & x_4'x_4 & x_4'y_4 & x_4'z_4 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \mathbf{0} \qquad \text{Eq. 15}$$

$$\text{with } p_{\alpha_i} = (x_i, y_i, z_i),\ p_{\beta_i} = (x_i', y_i', z_i'), \text{and } H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

≡ By substitution of A and h

$$Ah = 0 \qquad \text{Eq. 16}$$

The matrix h is known, however matrix A should still be found. In order to solve eq. 16 we use singular value decomposition (SVD) from eq. 17, which gives us the three matrices Σ , U and V. The matrices V_1 through V_9 represents the individual columns of V.

$$A = \Sigma \cdot \mathrm{U} \cdot \mathrm{V} \qquad \text{Eq. 17}$$

$$\text{with } \mathrm{V} = [V_1|V_2|V_3|V_4|V_5|V_6|V_7|V_8|V_9].$$

Now the last column of V gives us the solution for $A \cdot h$, this $h = V_9$. Details on SVD and how to find an appropriate scaling factor can be found in Zisserman (2004). We now know how to create a homography given four points we can explain the algorithm. When the algorithm is started the user should manually indicate four points on the screen that should be located in a square. The world coordinates points (0, 0), (0, 1), (1, 1) and (1, 0) are mapped to these points via H_w^0. The world points create a flat world wherein the length of the square is 1 unit length in the world (see also the world coordinate system in figure 19). For each new camera frame we search for corner points which are points with a high change in intensity. For more information on how to find them inside an image see section 4.1 "Corner detection". Corner points are then matched with points from the previous frame using a homographic projection. The projection that matches the most corner points is selected as the projection $\mathrm{H}_\mathrm{i}^{\mathrm{i}+1}$ that maps points from frame i to frame $\mathrm{i}+1$.

The method described in the paper of Simon et al. (2000) was implemented on a mobile device, however due to a number of limitations it was not used as a final solution for our problem. For example it suffers from drift; this was no surprise because it was already mentioned in Simon et al. (2000). For each successive homographic projection a small error is introduced which propagates in time and after a while the drift of the projected object becomes too large and we lose the augmented reality experience. Another obstacle was that in order to project an object we need to get from world coordinates to the last frame and are required to use all the homographic projections from the first to the last frame. This means that it is highly vulnerable for projection errors and if any of the homographic mappings are incorrect the projection of the object becomes wrong. It is crucial that the point matching algorithm always finds the correct mapping which in practice is not always the case. In our implementation the frame rate was just too low. The low frame rate resulted in wrong projections because of the relative large movement between frames.

3.3 Summary

In this chapter we have seen new developments and different approaches in order to solve the problems in the creation of AR. Most existing positioning systems like the one discussed in section 3.1.1 used GPS to determine location. However this was not an option for our system because GPS is not accurate enough and cannot be used in indoor environments. The method presented in section 3.1.4 used a stereo tracker to calculate the position of the endoscope. A stationary tracker would severely limit the usage of our mobile device and was not an option, however the usage of a tracker as mobile device was an option. Based on this observation a technique was implemented that combines visual localization with the use of Inertial measurements units from section 3.2.2.

4 APPROACH

A variety of techniques exists in order to calculate orientation and position. These methods range from using computer vision (e.g. camera) to GPS and inertial measurement units (IMU's). Our approach is based on using 3D point clouds which are captured by a stereo camera. These points are used to reconstruct a virtual world. The reconstruction of the world is based on point matching which is a relatively expensive operation in terms of processing time. By using the internal orientation sensors we calculate the orientation of the device. This orientation is used to speed up the point cloud matching process. Inside the virtual world we can project 3D models that are augmented with camera images. These models create the illusion of augmented reality and are displayed onscreen to the users. The three main tasks that should be performed are:

1. Detect corner points on both images captured from the stereo camera.
2. Generate a 3D point cloud from matches between corner points.
3. Reconstruct the transformation by performing point cloud registration.

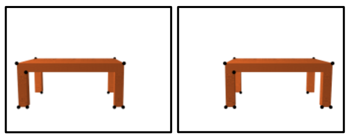


Figure 20 Corner detection

Figure 21 Cloud generation

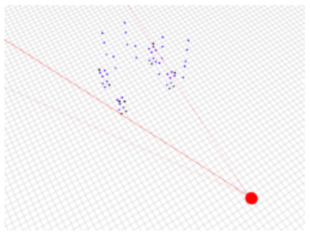


Figure 22 Cloud registration

The following sections will explain each of the above steps. In the final section we will discuss the implemented optimizations in order to increase performance in terms of accuracy and speed.

4.1 CORNER DETECTION

Corner detection is applied on the capture images that are retrieved from the stereo camera. It searches inside each image for corner points. The idea behind this approach is that for a reliable matching between points from the left and right stereo images the points with a high amount of information (e.g. corner points) are more interesting than points with a low entropy (for example on a large black wall). The specific corner detection we use is defined by Shi and Tomasi (Bradski & Kaehler, 2008, p. 318) (Shi & Tomasi, 1994). The following steps are performed:

1. We start by calculating the magnitude of the first order derivative of the image I. It is used to estimate the corner quality of a pixel. The x and y derivatives of the image are estimated with the Sobel operator. It uses 3×3 kernels that are convolved with the original image. For each pixel in the image derivatives we apply the autocorrelation matrix M. The matrix looks at the 3x3 surrounding neighborhood pixels $S(p)$ in the derivatives.

$$\mathrm{M} = \begin{bmatrix} \sum_{S(p)} \frac{dI}{dx}^2 & \sum_{S(p)} (\frac{dI}{dy}\frac{dI}{dx})^2 \\ \sum_{S(p)} (\frac{dI}{dx}\frac{dI}{dy})^2 & \sum_{S(p)} \frac{dI}{dy}^2 \end{bmatrix} \qquad \text{Eq. 18}$$

Now we compute the minimal and maximal eigenvalues of matrix M.

$$\lambda_1 = \frac{1}{2}(tr(M) - \sqrt{tr^2(M) - 4Det(M)}) \qquad \text{Eq. 19}$$
$$\lambda_2 = \frac{1}{2}(tr(M) + \sqrt{tr^2(M) - 4Det(M)})$$

With the pixel quality value as the minimal of the two eigenvalues $Quality = min(\lambda_1, \lambda_2)$.

2. Perform a 3x3 non-maxima suppression. For each pixel we look at the surrounding neighbors and compute the maximum quality value MaxQuality_{3x3}. We reject each pixel which has a quality value that is smaller than the maximum quality, $Quality < \text{MaxQuality}_{3x3}$. In this way we prevent the detection of corner points that are only locally maximum by thinning the edges.
3. Reject all corner points that have a quality value less than some pre-defined parameter and sort the remaining points in descending order.
4. Reject all corner points that have a quality value that is smaller than some pre-defined parameter α, $Quality < \alpha \cdot max(\lambda_1, \lambda_2)$.

4.2 Cloud Generation

We now have our corner points and need to match them from points in the left image to points in the right image. There are two types of correspondence estimation methods (Nalwa, 1993, p. 226). The first category compares the image intensity around two points and tries to find a match based on their similarity. The second category relies on edge detection and finds a match between points based on the correspondence between edges. We used a combination of both approaches. In the following section we will illustrate this by showing how corner points are matched based on their intensity calculated in section 4.2.2. After we found a match between corner points we can reconstruct its depth which is explained in section 4.2.4 and create a point cloud.

4.2.1 Point Matching

Corner point matching is done by finding for each point in the left image a match in the right image. In order to increase computation speed we narrowed the matching search space using epipolar geometry constraints. The epipolar geometry defines the geometry of stereo vision. It is used to impose constraints on the search space for point registration. For any two camera poses we can calculate the epipolar line, which describes the possible locations of a point when it is mapped from one camera to the other.

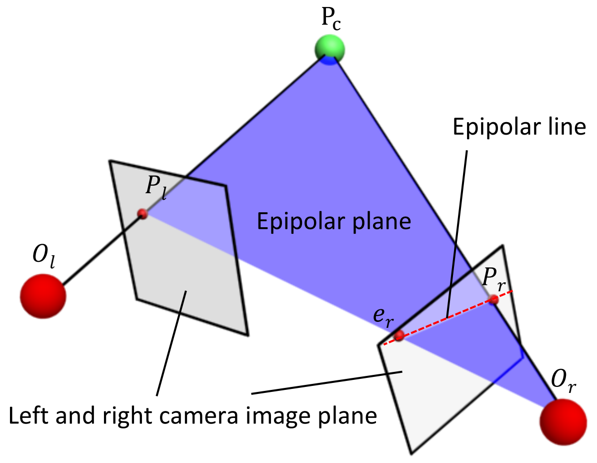


Figure 23 Two cameras with origins O_l, O_r and an epipolar line going through points e_r, P_r.

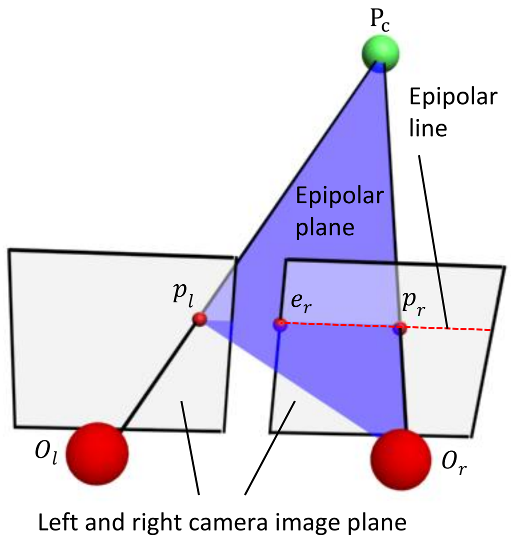


Figure 24 Two cameras with parallel image plains, creating in a horizontal epipolar line.

In figure 23 we can see the two camera origin points which are defined by O_l and O_r. On both image planes the point P_c is projected as P_l and P_r. Now the epipolar line is defined as the line between P_r and e_r, wherein e_r is the projection of point P_l on the right camera's image plane. In our setup we have a simplified situation because the image planes of the cameras are in parallel to each other. This creates horizontal epipolar lines as shown in figure 24. This means that a correspondence can only be found on the same scan line as the source point. So we match a corner point in the left image with a point in the right image only if they are on the same line. All possible corner points that are on the same horizontal line are collected and compared using the template matching approach explained in the next section. The point which is most equal (i.e. has the closest match) with the source point in the left image is considered as a match only if it is above a pre-defined threshold value.

4.2.2 TEMPLATE MATCHING

Template matching is a technique used to match a source image (the template) with a target image. The template should be smaller than the target because it should be fitted inside the target image. Let's define $T(x,y)$ as the pixel value of the template image at location (x,y) and $I(x,y)$, $R(x,y)$ to be the pixel value at the location (x,y) of the target resp. result image. The width and height of the template, target and result images are defined by T_w, T_h, I_w, I_h, R_w and R_h respectively. The size of these images are related as follows $R_w = T_w - I_w$ and $R_h = T_h - I_h$.

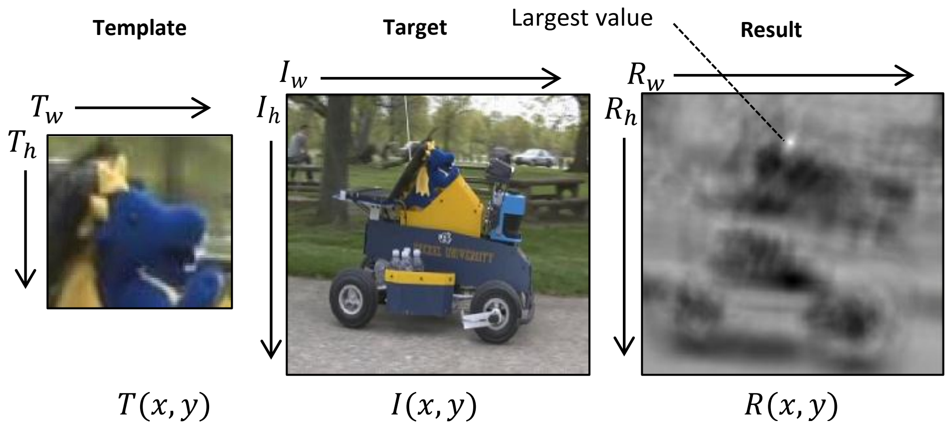


Figure 25 The two template and target input images and the result output image (Kuntz, 2009).

Now we can compare the template and target image given eq. 20 which is a normalized squared difference matching method between pixels.

Eq. 20

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') \cdot I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

The pixel $R(x,y)$ with the highest value is defined as the closest match between the template and target image.

4.2.3 CAMERA MATRIX

In order to project points in 3D space on our screen we need to use a camera model that maps 3D points to pixels coordinates. We use a well-known technique called the pinhole camera model. It uses a camera matrix (Zisserman, 2004, p. 163) in order to map 3D camera coordinates onto a 2D image plane which is translated into pixel coordinates.

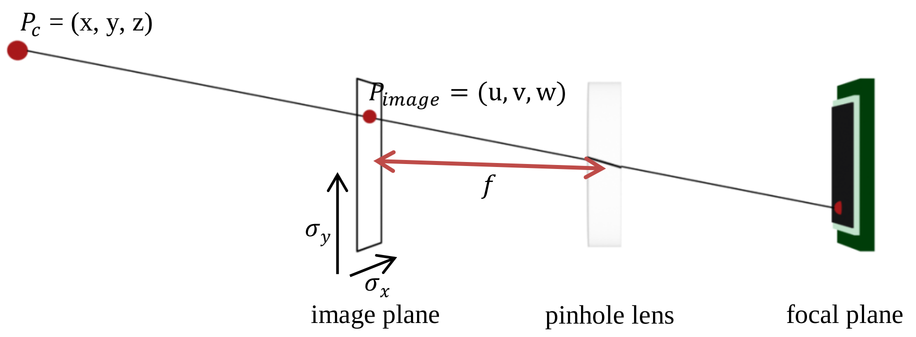


Figure 26 The relation between camera center and world coordinates.

The intrinsic camera matrix has the following components σ_x, σ_y that describe the coordinates of the image center in pixels; and focal lengths f_x, f_y that define the width and height of the focal plane in pixels. The focal length f of the camera describes the distance between the camera lens and focal plane in centimeters. The focal lengths have components $f_x = f \cdot m_x$ and $f_y = f \cdot m_y$ where m_x and m_y are scaling factors in pixels/centimeter for both the x and y direction of the focal plane. The matrix that maps a 3D camera point to the image plane is defined by,

$$M = \begin{bmatrix} f_x & 0 & \sigma_x \\ 0 & f_y & \sigma_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. 21}$$

From matrix M we can also calculate the generalized inverse matrix (also called the Moore-Penrose inverse) which maps points from the image plane back to camera coordinates, it is defined in eq. 22.

$$M^{-1} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{\sigma_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{\sigma_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. 22}$$

The image and camera coordinates are related to each other by,

$$P_{image} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = M \cdot P_c = \begin{bmatrix} f_x & 0 & \sigma_x \\ 0 & f_y & \sigma_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \text{Eq. 23}$$

$$P_c = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^{-1} \cdot P_{image} = \begin{bmatrix} f_x & 0 & \sigma_x \\ 0 & f_y & \sigma_y \\ 0 & 0 & 1 \end{bmatrix} P_{image} \qquad \text{Eq. 24}$$

Using the w component from the image coordinate P_{image} we can form a set of corresponding homogeneous coordinates. For example if we have the point (x, y) in Euclidean space then we have the following set of corresponding homogeneous coordinates $(x \cdot z, y \cdot z, z)$ with $z \neq 0$. So this single point (x, y) can be represented by infinitely many homogeneous coordinates. We can use these homogeneous coordinates in order to represent affine transformations (which are transformations that preserve straight lines). The image coordinate P_{image} is mapped to its corresponding homogeneous pixel coordinate by,

$$\frac{\mathrm{P_{image}}}{\mathrm{w}} = \begin{bmatrix} \mathrm{u/w} \\ \mathrm{v/w} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{x_{pix}} \\ \mathrm{y_{pix}} \\ 1 \end{bmatrix}$$ Eq. 25

Lefts define width and height to be the resolution in pixels of the camera. Based on this and eq. 25 we would expect that a pixel projected in the middle of the screen would have x and y coordinates equal to zero.

$$\begin{bmatrix} 0 \\ 0 \\ \mathrm{z} \end{bmatrix} \approx M^{-1} \cdot \mathrm{P_{image}} = M^{-1} \cdot \begin{bmatrix} \frac{\mathrm{width}}{2} \\ \frac{\mathrm{height}}{2} \\ 1 \end{bmatrix} \cdot \mathrm{z}$$ Eq. 26

In eq. 26 the approximately equal symbol is used because the matrix parameters are derived from calibration images suffer from noise. This noise introduces a small amount of error in the acquired camera parameters.

4.2.4 *Depth reconstruction*

Based on the intrinsic camera parameters derived in section 4.2.2 and the match between two corner points (section 4.2.1) we can calculate the distance of that point from the camera using triangulation. Let's assume a point P_c which is projected on the left and right image as P_l and P_r.

$$P_l = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, P_r = \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix}$$ Eq. 27

We define d to be the distance between both the cameras. The distance in centimeters α of P_c can be calculated as follows:

$$\frac{\mathrm{u} - \sigma_x}{f_x}\alpha + \mathrm{d} = \frac{\mathrm{u'} - \sigma'_x}{f'_x}\alpha$$ Eq. 28

The equation can be simplified because we assume that the intrinsic parameters for the two cameras are equal:

$$(\mathrm{u} - \sigma_x)\alpha + f_x\mathrm{d} = (\mathrm{u'} - \sigma_x)\alpha$$ Eq. 29

Now we derive $f_x\mathrm{d} = (\mathrm{u'} - \mathrm{u})\alpha$ in order to get the value for α:

$$\frac{f_x\mathrm{d}}{(\mathrm{u'} - \mathrm{u})} = \alpha$$ Eq. 30

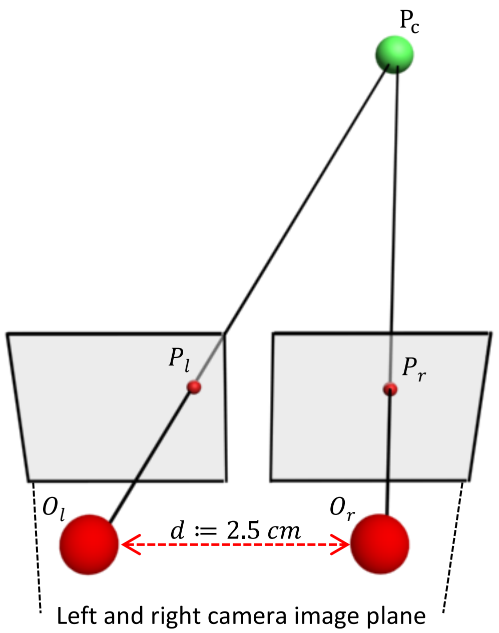


Figure 27 The stereo camera placed d centimeters apart with camera point $\mathrm{P_c}$ projected on both image planes.

Now the question is based on the two projections $\mathrm{P_l}$ and $\mathrm{P_r}$ of point $\mathrm{P_c}$, how far is it from the camera O_l? We know the distance between O_l and O_r which is 2.5 centimeters, this means that we can translate O_l in order to map it onto O_r using a translation matrix defined in eq. 1.

$$\mathrm{O_r} = \begin{bmatrix} & & & \mathrm{d} \\ & \mathrm{I} & & 0 \\ & & & 0 \\ & 0_3^T & & 1 \end{bmatrix} \mathrm{O_l} \qquad \text{Eq. 31}$$

Wherein d is the distance between the two camera lenses. Both $\mathrm{O_l}$ and $\mathrm{O_r}$ can be put into a translation matrix which results in the following equation.

$$\begin{bmatrix} \mathrm{I} & \mathrm{O_r} \\ 0_3^T & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \mathrm{O_l} \\ 0_3^T & 1 \end{bmatrix} \begin{bmatrix} & & & \mathrm{d} \\ & \mathrm{I} & & 0 \\ & & & 0 \\ & 0_3^T & & 1 \end{bmatrix} \qquad \text{Eq. 32}$$

We want to find the intersection between the two projection lines in figure 27. We know the pixel coordinates and the distance between the camera lenses and we need to find the depth in centimeters w''.

$$\begin{bmatrix} \mathrm{P_l} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{w''} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{w''} \cdot \mathrm{x_{pix}} \\ \mathrm{w''} \cdot \mathrm{y_{pix}} \\ \mathrm{w''} \\ 1 \end{bmatrix} \text{and} \begin{bmatrix} \mathrm{P_r} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{u'} \\ \mathrm{v'} \\ \mathrm{w''} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{w''} \cdot \mathrm{x'_{pix}} \\ \mathrm{w''} \cdot \mathrm{y'_{pix}} \\ \mathrm{w''} \\ 1 \end{bmatrix} \qquad \text{Eq. 33}$$

The following equation needs to be solved in order to get w''.

$$\begin{bmatrix} \mathrm{I} & \mathrm{O_l} \\ 0_3^T & 1 \end{bmatrix} M^{-1} \begin{bmatrix} \mathrm{P_l} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \mathrm{O_r} \\ 0_3^T & 1 \end{bmatrix} M^{-1} \begin{bmatrix} \mathrm{P_r} \\ 1 \end{bmatrix} \qquad \text{Eq. 34}$$

≡ By using eq. 32

$$\begin{bmatrix} \mathrm{I} & \mathrm{O_l} \\ 0_3^{\mathrm{T}} & 1 \end{bmatrix} M^{-1} \begin{bmatrix} \mathrm{P_l} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \mathrm{O_l} \\ 0_3^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} & \mathrm{d} \\ \mathrm{I} & 0 \\ & 0 \\ 0_3^{\mathrm{T}} & 1 \end{bmatrix} M^{-1} \begin{bmatrix} \mathrm{P_r} \\ 1 \end{bmatrix} \qquad \text{Eq. 35}$$

≡ By removing both translation matrices

$$M^{-1} \begin{bmatrix} \mathrm{P_l} \\ 1 \end{bmatrix} = \begin{bmatrix} & \mathrm{d} \\ \mathrm{I} & 0 \\ & 0 \\ 0_3^{\mathrm{T}} & 1 \end{bmatrix} M^{-1} \begin{bmatrix} \mathrm{P_r} \\ 1 \end{bmatrix} \qquad \text{Eq. 36}$$

≡ By resolving both matrix multiplications

$$\begin{bmatrix} \frac{\mathrm{w''} \cdot \mathrm{x_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} \\ \frac{v}{f_y} - \frac{\sigma_y w''}{f_y} \\ w'' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{d} + \frac{\mathrm{w''} \cdot \mathrm{x'_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} \\ \frac{v'}{f_y} - \frac{\sigma_y w''}{f_y} \\ w'' \\ 1 \end{bmatrix} \qquad \text{Eq. 37}$$

≡ By removing the last two rows which form a tautology

$$\begin{bmatrix} \frac{\mathrm{w''} \cdot \mathrm{x_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} \\ \frac{\mathrm{w''} \cdot \mathrm{y_{pix}}}{f_y} - \frac{\sigma_y w''}{f_y} \end{bmatrix} = \begin{bmatrix} \mathrm{d} + \frac{\mathrm{w''} \cdot \mathrm{x'_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} \\ \frac{\mathrm{w''} \cdot \mathrm{y'_{pix}}}{f_y} - \frac{\sigma_y w''}{f_y} \end{bmatrix} \qquad \text{Eq. 38}$$

≡ By removing the second row due to epipolar constraints we have $\mathrm{y_{pix}} = \mathrm{y'_{pix}}$ making it a tautology

$$\frac{\mathrm{w''} \cdot \mathrm{x_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} = \mathrm{d} + \frac{\mathrm{w''} \cdot \mathrm{x'_{pix}}}{f_x} - \frac{\sigma_x w''}{f_x} \qquad \text{Eq. 39}$$

≡ By multiplying with f_x

$$\mathrm{w''} \cdot \mathrm{x_{pix}} - \mathrm{w''} \cdot \mathrm{x'_{pix}} = \mathrm{d} \cdot f_x \qquad \text{Eq. 40}$$

≡ By solving $\mathrm{w''}$

$$\mathrm{w''} = \frac{\mathrm{d} \cdot f_x}{\mathrm{x_{pix}} - \mathrm{x'_{pix}}} \qquad \text{Eq. 41}$$

Using eq. 41 we now have a method to calculate the depth in centimeters of a projection.

4.3 CLOUD REGISTRATION

The goal of the cloud registration method is to find the best match between world points and new camera points that were just generated from the stereo images. By rotating and translating the camera points we try to fit them into the world and in this way we can reconstruct our translation and orientation in relation to the world.

Because there are infinitely many possibilities to match two point clouds we need to restrict our search space. A good idea is to use the available orientation sensors, with these sensors we can retrieve the orientation of the camera. The actual method of calculating orientation from sensors is explained in the next section. The only parameter we do not yet know is the translation of the camera points with respect to the world. The task of finding the correct translation is explained in the second section. We will conclude with point merging which is an optimization in order to decrease the number of points.

4.3.1 *Orientation*

The orientation of an object can be described by means of three orthogonal rotations; pitch, roll and yaw or sometimes also referred to as elevation, bank and heading.

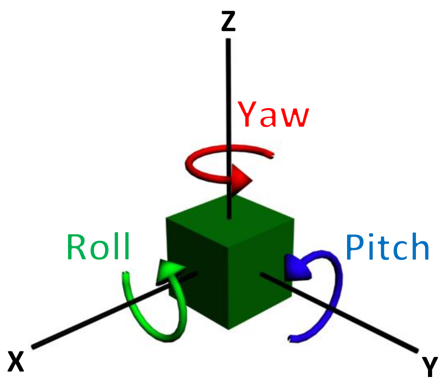


Figure 28 The orientation of an object can be described using; pitch, roll and yaw.

The orientation of our camera is calculated by means of acceleration meters, gyroscopes and a compass module. They need to work together in a specific way to correct measurement errors.

Accelerometers measure changes in acceleration, a value of 1 means that the sensor measures 1G (or 1 time the gravity). In order to conveniently parse the data the three sensors are aligned in each of the three dimensions. If the object would be in rest then we measure $(0, 0, 1)$ because we have the gravitation pull. It is possible to create a vector from these three values which can be used as an estimate to correct the pitch and roll of an object.

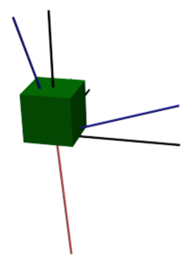


Figure 29 The acceleration vector in red.

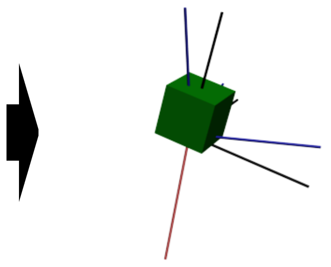


Figure 30 Inverse rotation matrix applied.

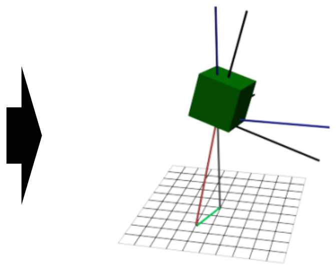


Figure 31 The error vector in green.

1. First we create an acceleration vector from the accelerometer values, this vector is described in an object coordinate system.
2. Rotate the acceleration vector with the inverse rotation matrix in order to go from an object to a world coordinate system.
3. The error vector can be used to correct the rotation matrix. We rotate the matrix in such a way that the length of this vector (and thus the error) is decreased. The ratio between the original length of the vector and the new length is defined by a predefined parameter.

The gyroscopes measure rotary changes and are also aligned in each of the three dimensions. A clockwise rotation results in a positive value and a counter clockwise rotation in a negative value. The speed of the rotation is represented by the magnitude of the value. Because the gyroscopes are aligned in each of the three dimensions we can measure changes in pitch, roll and yaw.

The compass module is needed in order to get the heading; it shows the derivation of magnetic north in clockwise direction (e.g. north is 0, east 90, south 180 and west 270 degrees). The actual implementation is less trivial because the magnetic module doesn't provide the actual heading and only gives the three values from the magnetic sensors inside the compass module. Like the accelerometers they measure the strength in each direction, only it is not gravity but the magnetic field. To get from these sensor values to the actual heading let's look at the following simplified 2D example.

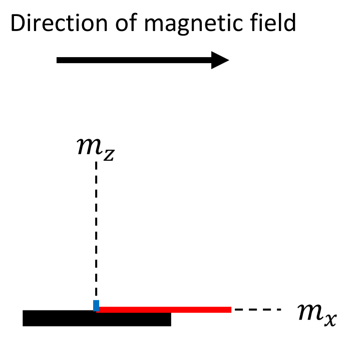


Figure 32 The compass module measuring magnetic fields in the m_x and m_z directions.

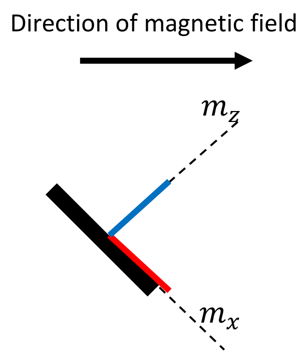


Figure 33 The same compass is tilted by 45 degrees, resulting in a changed measured magnetic field.

We can see how the two sensors change when they are rotated away from direction of the magnetic field. In our scenario we have three sensors, in a 3D situation the third sensor m_y would point away from the user (e.g. in the direction orthogonal to the magnetic field). If the compass is not tilted we can directly calculate the heading; we simply use eq. 42 and substitute $\hat{x}$ for x and $\hat{y}$ for y. However if the compass is tilted we should somehow compute these compensated values.

$$\tan2^{-1}(\hat{y},\hat{x}) = \begin{cases} \tan^{-1}(\hat{y}/\hat{x}) & x > 0 \\ \tan^{-1}(\hat{y}/\hat{x}) + \pi & \hat{y} \geq 0, \hat{x} < 0 \\ \tan^{-1}(\hat{y}/\hat{x}) - \pi & \hat{y} < 0, \hat{x} < 0 \\ +\pi/2 & \hat{y} > 0, \hat{x} = 0 \\ -\pi/2 & \hat{y} < 0, \hat{x} = 0 \\ \infty & \hat{y} = 0, \hat{x} = 0 \end{cases} \qquad \text{Eq. 42}$$

The m_x and m_y components can be compensated by translating $\mathrm{P_c} = [m_x \quad m_y \quad m_z]^T$ from camera to world coordinates. We can use eq. 4 and the rotation matrix we are updating in order to calculate $\Delta_{\mathrm{c}}^{\mathrm{w}} = (\Delta_{\mathrm{w}}^{\mathrm{c}})^{-1} = (\Delta_{\mathrm{w}}^{\mathrm{c}})^T$. Now we have the inverse rotation matrix and we can use it to translate $\mathrm{P_c}$ from camera to world coordinates.

$$\Delta_{\mathrm{c}}^{\mathrm{w}}\mathrm{P}_c = (\Delta_{\mathrm{w}}^{\mathrm{c}})^{-1}\mathrm{P}_c = (\Delta_{\mathrm{w}}^{\mathrm{c}})^T\mathrm{P}_c = \Delta_{\mathrm{c}}^{\mathrm{w}}\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} = \mathrm{P_w} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ m_{z'} \end{bmatrix} \qquad \text{Eq. 43}$$

The heading can now be calculated by using eq. 43 and eq. 42. Note that in the above equations the computed $m_{z'}$ values are not relevant because it describes the rotation of the z component which is not used in our heading calculations.

A rotation matrix is needed in order to describe the orientation of the camera. On each iteration we update the rotation matrix with changes measured by the gyroscopes. These updates should be performed in the order of kHz to be useful. These sensors are not 100% accurate and each measurement will introduce a small error in the orientation update. To compensate for this we correct the rotation matrix with the actual heading measured by the compass and pitch/yaw from the accelerometer. In order to let this approach work we have to assume that on average the accelerometers will point towards the ground (which is introduced by the gravitationally pull) and the compass points towards north.

4.3.2 *TRANSLATION*

Now we know the orientation of our cloud w.r.t the world we only need to find the translation. We need to define some quality measure in order to determine the score of a match between camera and world points. We propose the following quality measure;

$$Score(C,W,\varepsilon) = \sum\nolimits_{w \in W} \max_{c \in C}(((1 - \min\left(1, {}^{\mathrm{d}(w,c)}/_{\varepsilon}\right)) \frac{c_s w_s}{2}) \qquad \text{Eq. 44}$$

With Euclidean distance $\mathrm{d}(p,p') = \sqrt{(p_x - p'_x)^2 + (p_y - p'_y)^2 + (p_z - p'_z)^2}$,
Camera point cloud C and World point cloud W.

The goodness or score between two point sets is defined by their separation in Euclidian space and strength value. For each point p we have a strength parameter p_s that defines how sure we are that a point is present. The initial strength is derived from the similarity value of the template matching algorithm (section 4.2.2).

For each world point w we find the highest scoring (thus nearest) camera point in the other set. If the distance is smaller than 1 we take that distance and divide it by ε to increase the total score value. This means that if the distance between both points is zero the score is maximal. On the other hand if the distance is larger than ε we don't care and just add a score of 0. In this way we try not to search for an *average* maximal score but for an error which is minimal when the *greatest number* of individual point matches are found. We further multiply this value by the average strength values to reflect point strength in the score result.

$$NormScore(C,W,\varepsilon) = \frac{Score(C,W,\varepsilon)}{|W|} \qquad \text{Eq. 45}$$

We now normalize the score value w.r.t. the number of world points. Using this normalized score we finally want to find a translation $t \coloneqq P_w - P_c$ of C such that the score we have is maximized.

$$\mathrm{Match(C,W,\varepsilon) = max(P_c \in C \wedge P_w \in W \wedge C_t \coloneqq Tran(C, P_w - P_c) \colon NormScore(C_t, W, \varepsilon) \colon C_t)} \qquad \text{Eq. 46}$$

With $\mathrm{Tran(P,t)}$ the set of translated points w.r.t. translation t and set P

The idea is to match the camera points C with the world W by searching for the smallest error given a translation t for the camera points; this translation translates the camera points from camera to world perspective. The possible translations we take into account are the ones that map a given point in C to a point in the world W, thus $P_w - P_c$. Because this is an exhaustive search we could optimize it by only using a subset C_{sub} of constant size. Taking a subset of world points would not be a good idea because for each point we expect it has an equal possibility to match it with the world, however the other way around is not necessarily valid and thus we should take all world points into account. This results in a matching approach which is dependent on the number of world points.

4.4 Optimizations

We will now discuss implemented optimizations which have as effect an increased frame rate and higher tolerance to noise. The depth scaling technique deals with the fact that in general the calculated depth error of a 3D point in a camera coordinate system is larger than the error in the x and y direction. In the second optimization technique we try to decrease the number of points inside the point clouds. In this way we gain frame rate because there are less points to compare during the score calculation.

4.4.1 Scaling

Instead of directly applying the score function from eq. 45 to a world and camera point cloud, we first scale it to increase the tolerance for measured depth errors. The scaling should be done in camera coordinates because the depth is measured in the perspective of the camera. Thus we first transform the world point cloud to the camera coordinates using $\Delta_c^w W_w = W_c$. In figure 34 we have the camera with camera point cloud C_c and world point cloud W_c. The camera points C_c of world point w_c are likely to be projected along the homogenous coordinates that point.

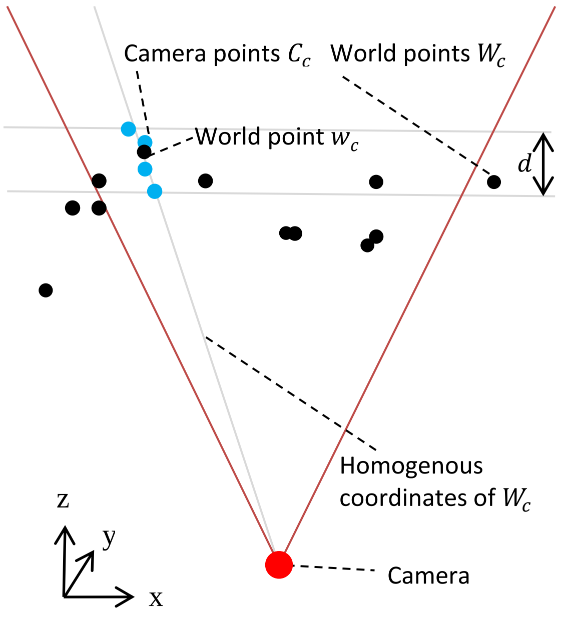


Figure 34 Camera with world and camera point clouds.

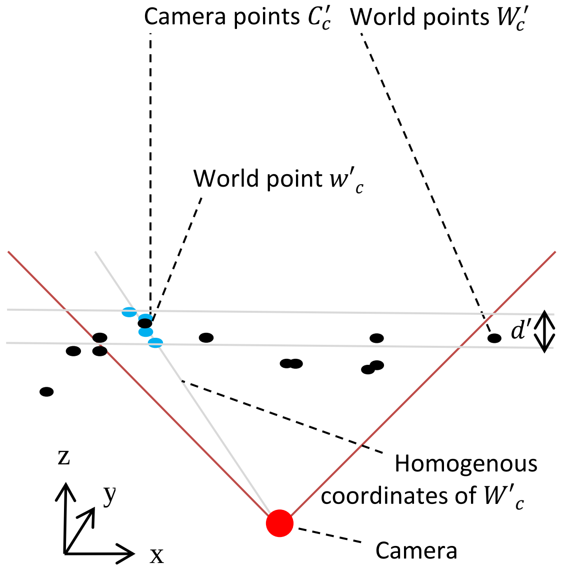


Figure 35 The same camera with scaling applied.

In figure 35 we have the world and camera point clouds after scaling is applied. The scaling factor is defined by α, thus $d \cdot \alpha = d'$. The scaling on the z component is performed using the matrix S_α in eq. 47.

$$S_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq. 47}$$

The scale factor α determines the amount of error in the x and y components with respect to the depth. A smaller scale factor would produce a higher matching score even if the distortion in the depth is larger. The consecutive matrix multiplications for the world and camera point clouds are shown in eq. 48 and eq. 49.

$$S_\alpha \Delta_c^w W_w = W'_c \quad \text{Eq. 48}$$

$$\mathrm{S}_{\alpha}\mathrm{C}_{c} = \mathrm{C'}_{\mathrm{c}} \qquad \text{Eq. 49}$$

The score function from eq. 45 can now be applied to the scaled world $\mathrm{W'}_{w}$ and camera $\mathrm{C'}_{c}$ point clouds. In order to transform the clouds back we can use eq. 50 and eq. 51.

$$\Delta_{\mathrm{w}}^{\mathrm{c}}\mathrm{S}_{1/\alpha}\mathrm{W'}_{w} = \mathrm{W''}_{\mathrm{w}} \qquad \text{Eq. 50}$$

$$\mathrm{S}_{1/\alpha}\mathrm{C'}_{c} = \mathrm{C'}_{\mathrm{c}} \qquad \text{Eq. 51}$$

4.4.2 *MERGING*

To increase performance we try to optimize a single point cloud by merging individual points together. When two points p and $\mathrm{p'}$ are separated from each other by a distance smaller than some pre-defined constant α, e.g. $\mathrm{d}(p, p') < \alpha$. Given the two strength values p_{s} and $\mathrm{p'}_{s}$ the merged point strength would be;

$$\mathrm{p''}_{s} \coloneqq \mathrm{p}_{s} + \mathrm{p'}_{s} - \mathrm{p}_{s}\mathrm{p'}_{s} \qquad \text{Eq. 52}$$

This is due to mutually non-exclusiveness of the two events (Montgomery & Runger, 2011, p. 38).

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \qquad \text{Eq. 53}$$

Thus the probability that events A and B occur is equal to the probability that event A or event B happens minus the possibility that both events happen. Besides the likelihood of a point also the coordinates should be merged. In our implementation the point with the highest strength has the most influence on the merged values, it is achieved using the following fractions; $frac_{p}^{p'} \coloneqq \frac{\mathrm{p}_{s}}{\mathrm{p}_{s}+\mathrm{p'}_{s}}$ and $frac_{p'}^{p} \coloneqq 1 - frac_{p}^{p'}$. The corresponding x, y and z coordinates are updated as follows;

$$\begin{aligned} \mathrm{p''}_{x} &\coloneqq \mathrm{p}_{x} frac_{p}^{p'} + \mathrm{p'}_{x} frac_{p'}^{p} \qquad \text{Eq. 54} \\ \mathrm{p''}_{y} &\coloneqq \mathrm{p}_{y} frac_{p}^{p'} + \mathrm{p'}_{y} frac_{p'}^{p} \\ \mathrm{p''}_{z} &\coloneqq \mathrm{p}_{z} frac_{p}^{p'} + \mathrm{p'}_{z} frac_{p'}^{p} \end{aligned}$$

5 IMPLEMENTATION

In the next section we will discuss the development model used during this project, in the subsequent section the different specifications of the mobile device will be explained. The remainder of this chapter will discuss the application that was developed.

5.1 DEVELOPMENT MODEL

Different techniques and methods exist in order to manage deliverables and develop software. The software development methodology used in this project is scrum. This method is incremental and Agile based, in figure 36 we can see the Scrum approach (Kniberg, 2007).

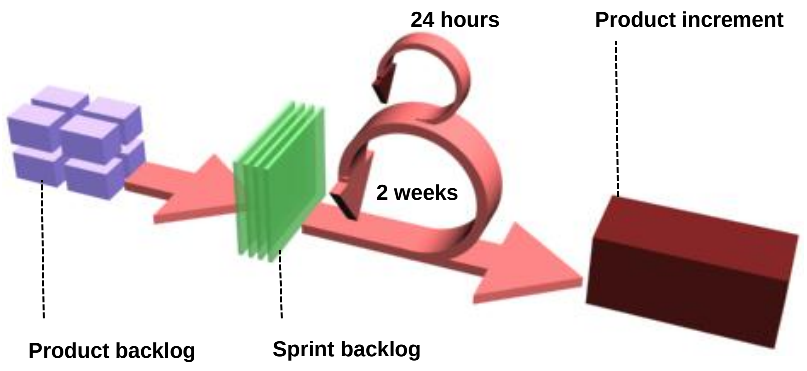


Figure 36 The Scrum development model with product increment cycles.

The Scrum method has different roles; we have the scrum master, product owner and team. The scrum master is responsible for maintaining the processes and coaching his team. He can be thought of as the project manager. The product owner represents the business or costumer and guides the team toward building the right product. The team is the group of people who perform the actual work on the product (e.g. design and implementation).

The product requirements are listed in the product backlog and functions as a list of features that should be implemented. They are ordered and executed based on their importance. The project progresses during these sprints, which usually take about two weeks. A sprint is defined as a unity of work that is executed by the team. Before the sprint starts features are taken from the product backlog and stored in the sprint backlog. Features inside the sprint backlog should be finished during a sprint. However, if for some reason a sprint backlog item is not completed before the sprint finishes it is put back into the product backlog. After the sprint is finished a sprint review meeting is held, during this meeting the team demonstrates the newly integrated features to the product owner. The product owner in its turn provides feedback to the team. During this meeting the goals and features for the next sprint are collected which are often influenced by the comments from the product owner.

In this specific project there is a project team which consists of a single person, T.K.A. (Thorstin) Crijns. The product owner is Alten and represented by Ir. D.W. (David) Hardy and Drs. A.A.G. (Ard) Willems.

5.2 MOBILE DEVICE

In this section we will describe the development environment that was used along with its dependencies. We will continue with the target environment of the mobile device and conclude with the different used hardware components.

5.2.1 DEVELOPMENT ENVIRONMENT

The application was developed using the Eclipse IDE for Java Developers, it was targeted for the Android platform version 2.3.3 and compatible with the API level 10. Using the Real3D Add-On we get an actual 3D application meaning that it can use the 3D screen and stereo camera. Because of the required image processing techniques the OpenCV (version 2.3.1) computer vision library is used.

5.2.2 TARGET ENVIRONMENT

The target environment is the LG Optimus 3D P920 which is an Android based smartphone. In order to use the most recent API functionalities the OS was upgraded to the latest version available at the time of development, called Gingerbread (version 2.3).

5.2.3 CAMERA CALIBRATION

In order to get the intrinsic camera parameters calibration was performed on the stereo camera. We used a software tool "Camera Calibration Tools" (2012) and calibrated each of the two cameras separately. For the calibration the following pattern was used.

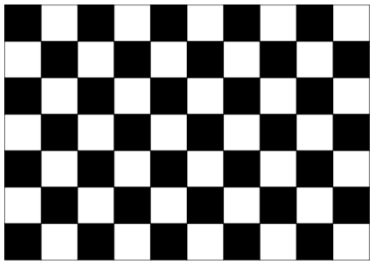
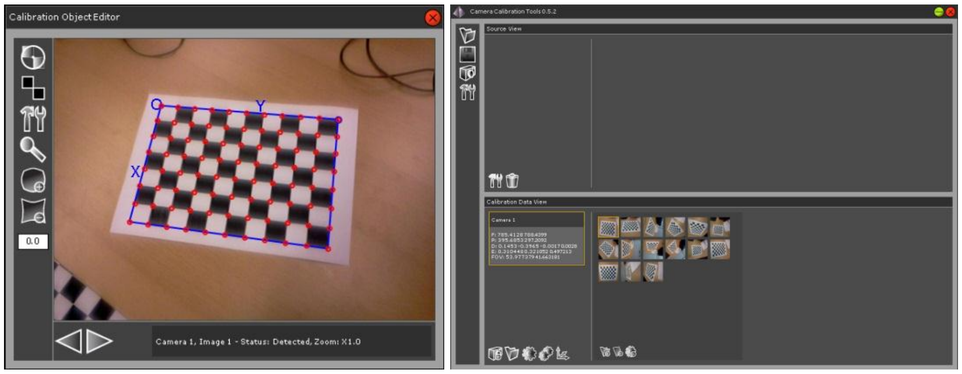


Figure 37 The used calibration pattern was a 7x10 checker pattern.

The calibration pattern was photographed from different angles at a resolution of 320 by 480 pixels. On each image the pattern was manually marked as shown below, in total there were 15 calibration picture (see figure 38).


Calibration Object Editor
O
Y
X
0.0
Camera 1, Image 1 - Status: Detected, Zoom: X1.0
Camera Calibration Tools 0.5.2
Source View
Calibration Data View
Camera 1

F: 785.4126 788.4399
P: 395.6853 297.2092
D: 0.1453 -0.3965 -0.0017 0.0026
E: 0.310448 0.221052 0.497213
FOV: 53.977379 41.663191

Figure 38 For each image the checkerboard corners were manually marked.

Based on the calibration images the relative transformations and checkerboard were calculated as visualized in figure 38. In the left picture we can see the relative transformations from a camera coordinate system, on the right the same mapping but from a checkerboard coordinate system.

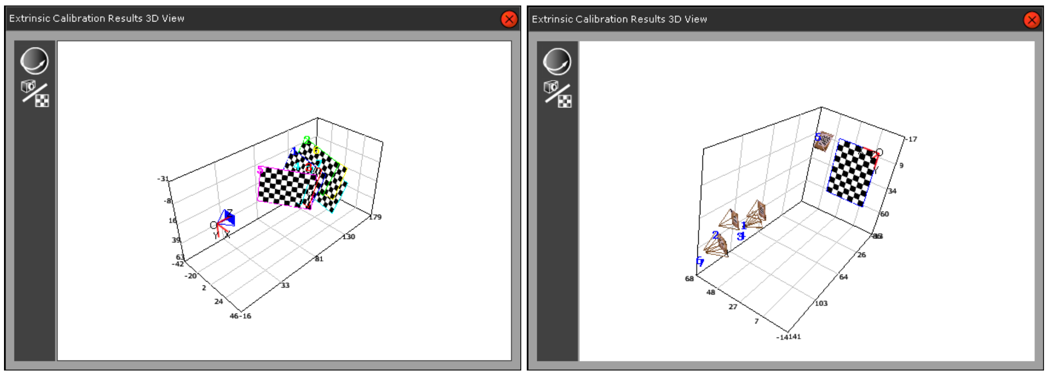

Extrinsic Calibration Results 3D View
Extrinsic Calibration Results 3D View

Figure 39 The computed relative positions of the camera and checkerboard.

Based on all these images the camera positions, extrinsic and extrinsic parameters were calculated, we have the following intrinsic parameters for the left camera.

$$f_x \coloneqq 833,\, f_y \coloneqq 840,\, \sigma_x \coloneqq 182,\, \sigma_y \coloneqq 229 \qquad \text{Eq. 55}$$

Using eq. 55 It is now possible to substitute these values in the matrix from eq. 21 resulting in the actual calibration matrix for our specific camera.

$$\mathrm{M}_l \coloneqq \begin{bmatrix} 833 & 0 & 182 & 0 \\ 0 & 840 & 229 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{Eq. 56}$$

The same steps are performed for the right camera which provides us with the intrinsic parameters in eq. 57.

$$f'_x \coloneqq 880,\, f'_y \coloneqq 874,\, \sigma'_x \coloneqq 185,\, \sigma'_y \coloneqq 216 \qquad \text{Eq. 57}$$

Based on the parameters the calibration matrix in eq. 58 was constructed.

$$\mathrm{M}_r \coloneqq \begin{bmatrix} 880 & 0 & 185 & 0 \\ 0 & 874 & 216 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{Eq. 58}$$

In order to simplify the calculations that are performed in the subsequent sections we assume both matrices are equal, e.g. $\mathrm{M} = \mathrm{M}_l$ and $\mathrm{M}_r \coloneqq \mathrm{M}_l$. By doing so we increase computation speed at the cost of accuracy.

5.2.4 *Hardware*

The LG Optimus has the following hardware components; a GPS module, three accelerometers, three gyroscopes, compass, speakers, microphone, stereo camera and 3D touchscreen. The display is 4,3 inch and has a resolution of 800 x 480 pixels, it is possible to enable stereo vision by simultaneously projecting two images onto the screen which give the illusion of 3D.

Figure 40 The LG Optimus 3D with stereoscopic camera (Toor, 2011).

In figure 40 we can see the stereoscopic camera, it is a 5 megapixel camera and able to capture stereo images. It should be noted that for our application not all components were needed, e.g. the GPS module, speakers and microphone remained unused.

5.3 Application

In the first part of this section we will explain the structure of the application and look at the different software components. We will also examine the different application modes and show we can switch between them. Different data files are created during the execution of the application. The last section will show how they are created and explain the purpose of each file.

5.3.1 Design

An efficient and modular design is essential in order to improve performance and decrease maintenance costs. The different threads create a more modular architecture and decrease the overall computation time by utilizing the multicore processor. In figure 41 a global overview is shown of the different software components.

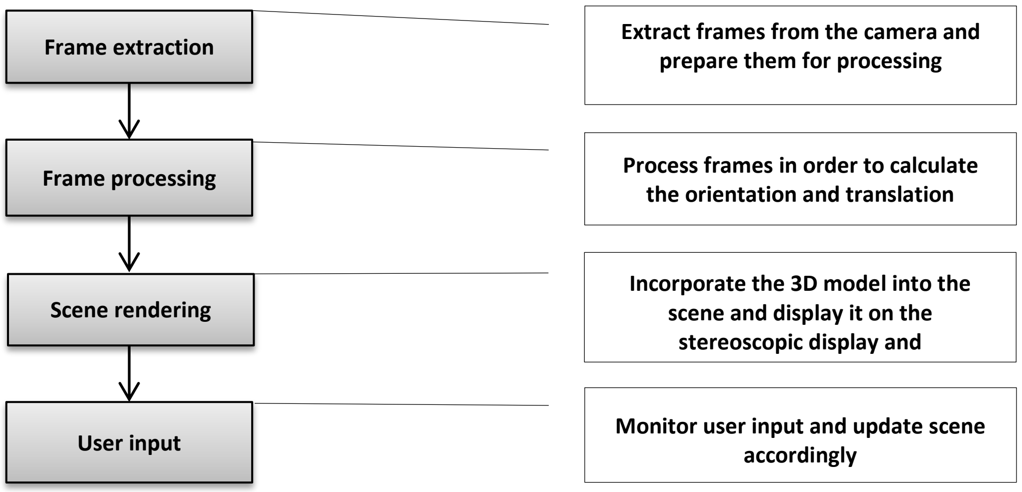

Frame extraction
Extract frames from the camera and prepare them for processing
Frame processing
Process frames in order to calculate the orientation and translation
Scene rendering
Incorporate the 3D model into the scene and display it on the stereoscopic display and
User input
Monitor user input and update scene accordingly

Figure 41 The four main components and dependencies between them.

The frame extraction module handles the data processing for each stereo frame that is received from the camera. In order to split the stereo data we first need to translate it into the appropriate data format. The raw data from the camera is in YUV420 (also known as NV21) and should be converted to ARGB. Image conversion is performed by converting each pixel of the image which has three components. The first component is the Y signal that represents the luminance. The other two U and V components are the blue and red luma values; these are defined by resp. the blue and red color without luminance. In NV21 these components are encoded as defined in (Bouguet, p. 10), for each component one byte is used, these bytes are ordered as shown in figure 42.

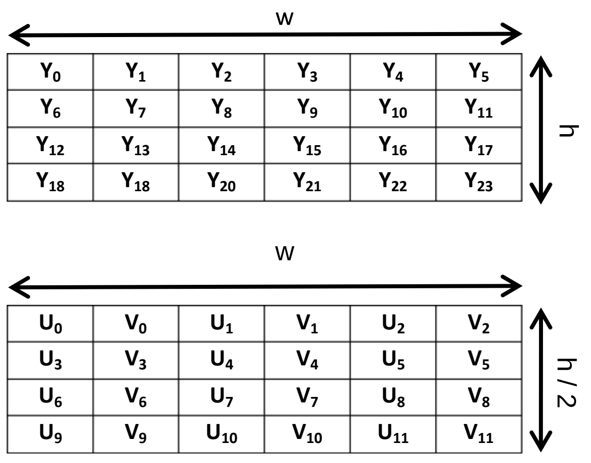


Figure 42 The two data blocks of an $w \times h$ image in YUV420 format.

The first $w \cdot h$ bytes in the raw data stream of the above image are the luminance components which represent a gray scale image. The blue and red luma values can be found in the subsequent $w \cdot \frac{h}{2}$ bytes, they form a 2×2 subsampling of the original image. In total the data stream has a size of $\frac{3}{2} h \cdot w$ bytes creating an $w \times h$ YUV420 image. In figure 47 the mapping between the Y and U/V components is shown, they form an interleaved U/V plane.

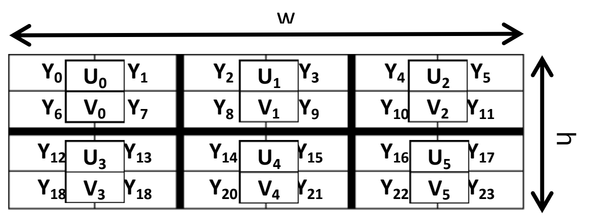

w
Y0 U0 Y1 Y2 U1 Y3 Y4 U2 Y5
Y6 V0 Y7 Y8 V1 Y9 Y10 V2 Y11
Y12 U3 Y13 Y14 U4 Y15 Y16 U5 Y17
Y18 V3 Y18 Y20 V4 Y21 Y22 V5 Y23
h

Figure 43 A $w \times h$ image has the following mapping between blue/red and luminance.

The ARGB8888 bitmap has four components; alpha, red, blue and green. Each component is one byte, thus in order to represent an $\mathrm{w} \times h$ image we need $4 \cdot \mathrm{w} \cdot \mathrm{h}$ which form a 24-bit true color image with an 8-bit alpha channel. The alpha value 0x00 denotes a fully transparent pixel and a 0xFF a fully opaque pixel. Bytes are ordered as shown in figure 44.

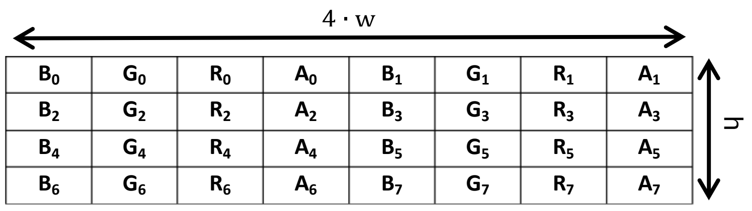


Figure 44 A true color ARGB8888 image of size $w \times h$ is represented by $w \cdot h$ 32-bit integers.

By applying the matrix transformation in eq. 59 we can convert an image in YUV420 format to ARGB8888.

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \\ \mathrm{A} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.5806 \\ 1 & 2.03211 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$ Eq. 59

The pseudo inverse matrix in eq. 60 gives the mapping from ARGB888 to YUV420. It should be noted that we lose the alpha value because this is not supported in the YUV420 image format.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299002 & 0.586999 & 0.113999 & 0 \\ -0.147139 & -0.288862 & 0.436 & 0 \\ 0.615002 & -0.514988 & -0.100014 & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ A \end{bmatrix}$$ Eq. 60

After the camera data is transformed into a bitmap we get an image of 640 by 480 pixels (we used the lowest camera resolution to increase processing time). The bitmap contains both stereo images from the camera as shown in figure 45.

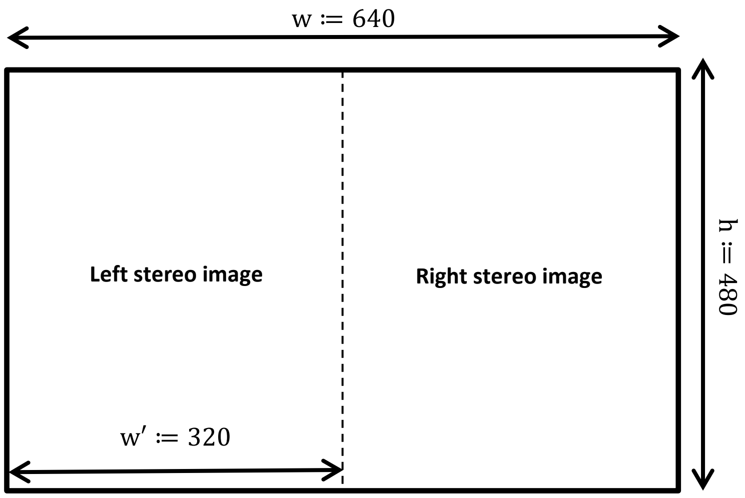


Figure 45 The left and right stereo image of size $w' \times h$ are split from the original $w \times h$ image.

The retrieved images are from the left and right camera and have a size of 320 by 480 pixels. Both of these images are passed onto the frame processing module.

The frame processing module estimates the location of the camera based on the stereo images, orientation and intrinsic camera parameters. A separate thread handles the orientation by continuously pulling data from the sensors and updating the rotation matrix as described in section 4.3.1. Based on the stereo images a point cloud is generated which is matched with the world point cloud, the matching algorithm returns the translation in order to get from camera to world coordinates. Because the location of the 3D model is known based on the translation we can calculate the location and orientation of the object with respect to the camera.

The calculated location and orientation of the 3D model with respect to the camera is used by the scene rendering module to project the model over the scene (bitmap) from the camera image. In order to save processing time scene updates are only performed when a new camera image is received or the location of the model changes.

User input is handled by the user input module. It processes GUI commands and is exclusively invoked by events (for example if the user wants to view the context menu to switch to a different application mode).

There are not only dependencies between software modules but also between modules and the different hardware components, they are visualized in figure 46.

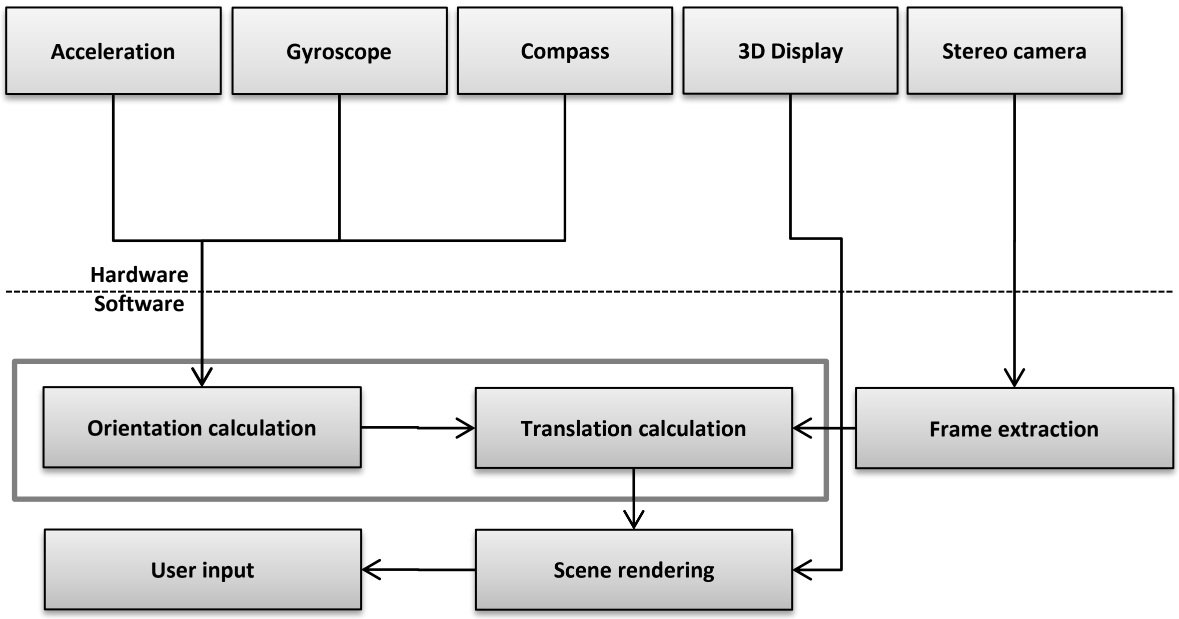


Figure 46 The dependencies between hardware components and software modules.

In the above model the dataflow and dependencies between the different hardware components and software modules are indicated by arrows. An arrow from component A to B indicates that B depends on A. The frame processing module from figure 46 is split into its two sub modules (e.g. the orientation and translation calculation modules). The orientation module requires the gyro's accelerometers and compass module in order to determine orientation. The translation module does not depend on any hardware components, however it requires the frame extraction module which in its turn uses the stereo camera.

5.3.2 *Modes*

In this section we describe the different application modes that the application has. In figure 47 all modes and flows between them are visualized.

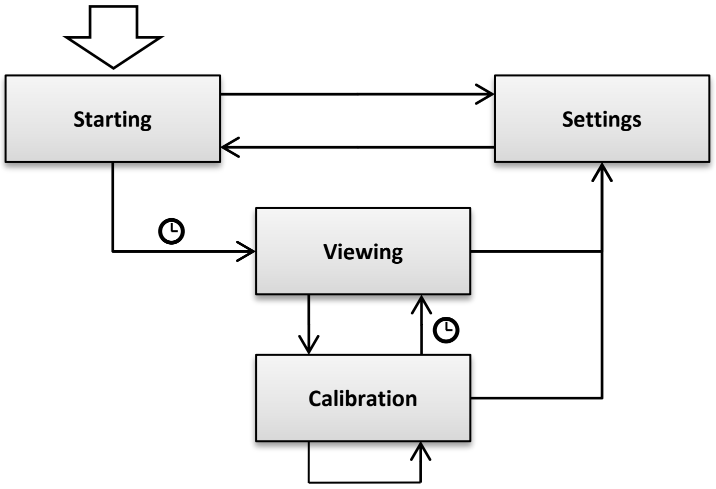


Figure 47 The different application modes.

In the above flow diagram the clock symbol indicates that the event is triggered automatically. An arrow without a symbol means that it can only be triggered manually by the user. When the application is initialized we enter the starting mode. It waits for the orientation module until it reports that an accurate orientation has been acquired and enters the viewing mode. The orientation is defined to be accurate when the orientation error measured by the compass and accelerometers is below a predefined threshold value. If the user triggers the context menu and enters the settings page the application will exit the starting mode and enter the settings mode. The settings mode is used to modify application parameters such as the point cloud merge distance, number of world points and maximum allowed template difference. The user can change settings which are automatically saved to the settings file. If the menu is closed the startup process is activated. The viewing mode tries to find the orientation and translation of the camera in relation to the 3D model which is then drawn into the scene. The user can enter the calibration mode or change settings in the settings page. If the calibration mode is active then the user can create the world point cloud and location of the 3D model. A point cloud of the world is built from the camera images, which represents the area currently visible to the camera. This environment is the world space which consists of 3D points (the point cloud). If the environment is reliable enough (e.g. satisfies the constraints which are set in the settings page) then it automatically stores the world space and triggers the startup process. However the user can also manually exit via the settings page.

5.3.3 DATA FILES

The settings file stores all the parameters of the application, it is a binary file consisting of integer values and is saved under the name “settings" in the root folder. All the application settings are categorized into 4 groups; settings in relation to world camera matching, corner points, world points and point strength. For each category the parameters are summarized in the tables below. The data types shows the expected data type and is either in mm, cm, *, #, %. The ‘mm’/’cm’ represents a unit length in millimeters/centimeters, the factor ’*’ is a multiplication factor, ‘#’ is the number of elements and ‘%’ defines the percentage which should usually be in the range of 0-100.

Index	Description	Type
1	How large is the error in depth projection in relation to the other directions. If this value is 1 it means that depth error is equal to the error in the other directions.	*
3	The minimal acceptable point cloud matching score.	%
7	How tolerant should we be to errors when in viewing mode. A higher value results in more matching but also in more false positives. If this factor is 1 it means we are as tolerant for errors in calibration mode as in viewing mode.	*

Table 2 World camera matching settings.

Index	Description	Type
2	Minimal number of corner points needed in order to continue with the generation of a camera point cloud.	#
4	The maximal number of corner points that can be detected by the corner point algorithm. The value represents corner points on the left in addition to the right camera.	#
5	The minimal required corner strength.	%
11	The maximal acceptable template difference.	%
15	The maximum allowed depth of a point (a larger depth results in greater errors).	cm

Table 3 corner points settings.

Index	Description	Type
6	The maximum number of points that we can store in the world.	#
10	The required number of world points when calibration should be stopped.	#

Table 4 World points settings.

Index	Description	Type
8	Maximal acceptable error between camera and world points when matching points are earned.	mm
9	The distance when points should be merged together.	mm
12	The initial point strength which is multiplied by the template score.	%
13	Points that have a lower strength than this value will be removed and are classified meaningless.	%
14	The percentage of remaining point strength at each calibration iteration.	%

Table 5 Point strength settings.

In addition to the settings file there is also a file for storing information about the calibrated environment. When construction of the world point cloud is finished all information about the points and location of the model are saved in the world file. It is located in the root directory under the name ‘world’. These world files can be swapped between mobile devices to exchange environment models. The data format is binary, however there is a separate file with the extension ‘xyz’ that can be read by many 3D point cloud visualization applications including CloudCompare (2010). It is an open source initiative that can be freely downloaded.

6 RESULTS

In this chapter we will perform various tests in order to determine the capabilities of the stereo camera. We will conclude this chapter by summarizing our findings and look at possible improvements.

6.1 VALIDATION

The different tests are executed using the test cube in figure 48. The cube can be used to verify the results because it has known spatial and visual properties. It has a high contrast with black points which are detected by the camera as corner points.

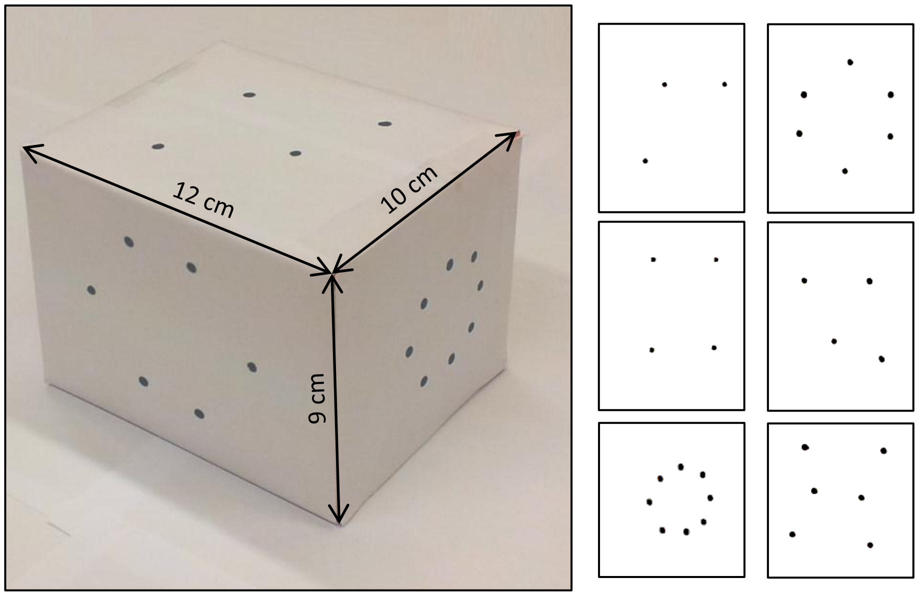


Figure 48 The test cube of 9x10x12 cm with point patterns on each side.

In total we will perform 5 tests. Each test is designed to asses different properties of the system. In the first test we will simply track one single point and determine the accuracy of the measured depth. The second test is used to determine the accuracy of the generated point cloud. In the third test we will measure the accuracy of the detected camera location in relation to the 3D model by comparing the measured location to the known real location of the camera. In the corner point test we will look at the impact of incorrectly detected corner points. Finally in the last test we measure timing statistics for each of the different processing tasks.

6.1.1 DEPTH TEST

In this test we will verify the accuracy of the calculated depth by the stereo camera. We have a single black corner point that is drawn on a white surface. This point is captured by the stereo camera that will create a point cloud of just a single point. The data we collect is the depth of this point cloud. In figure 49 we can see the test setup.

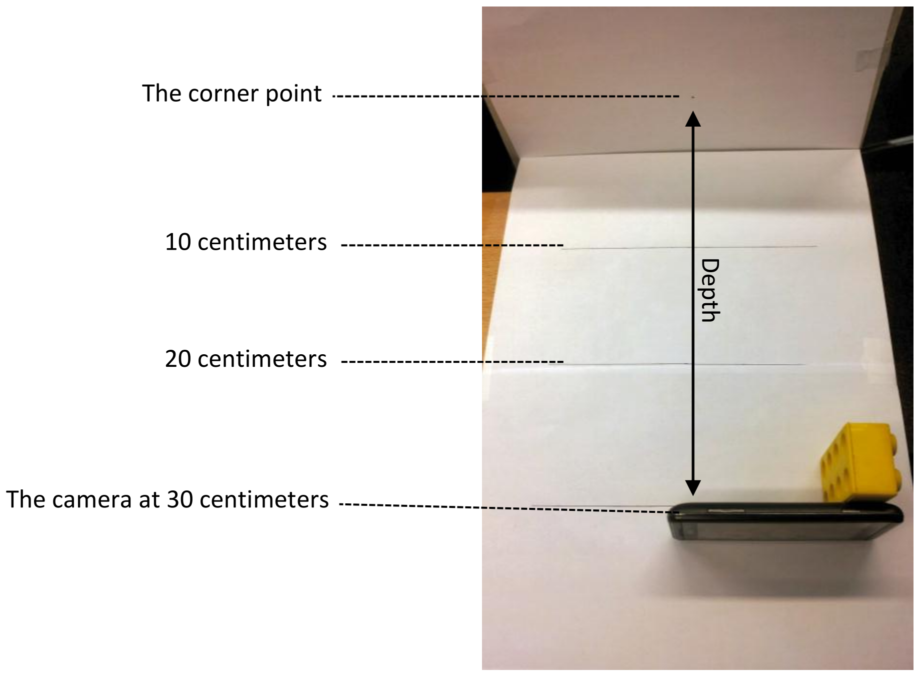
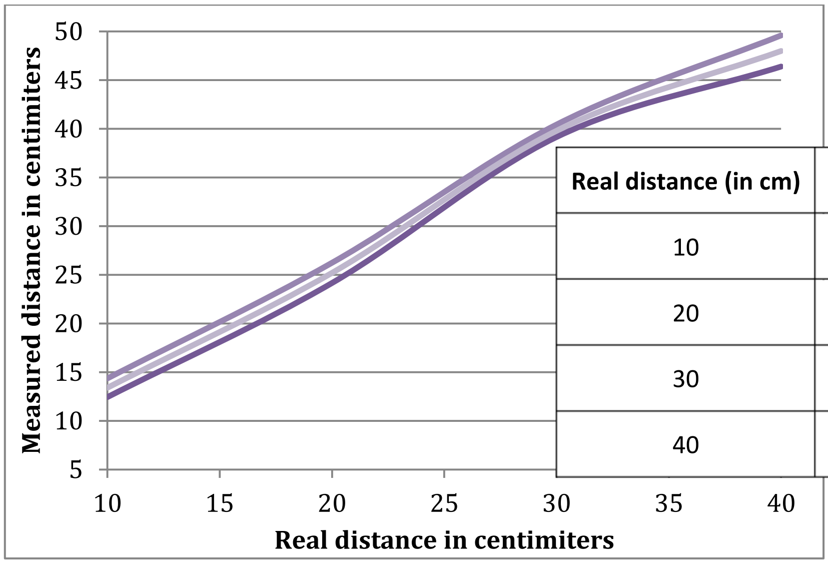


Figure 49 The depth test.

The test is executed a number of times for different distances. In plot 1 we performed 35 measurements over the distances of 10, 20, 30 and 40 centimeters.



Real distance (in cm)	#	Mean (in cm)	SD (in cm)
10	10	13,4	0,96
20	10	25,2	1,04
30	10	39,8	0,64
40	15	48	1,6

Plot 1 The measured distance compared to the real distance with standard deviations

In the plot we have the real distance on the x-axis and the measured distances on the y-axis which are all in centimeters. The two lines above and below the average measured distance are the standard deviations, we can see they are minimal for a distance of 30 centimeters. The measured and real distances are off by a factor of 1.2 which could be the result of a non-perfect alignment of the two stereo cameras or small errors in the calculated camera parameters during calibration.

6.1.2 PROJECTION TEST

In this test we will compare the generated point cloud with the real point on the test cube. The camera was targeted at one side of the cube in order to generate a world point cloud. The cloud (stored in the xyz file, see section 5.3.3) was copied and opened in CloudCompare (2010). The point cloud together with the test cube resulted in the following projections.

● Camera

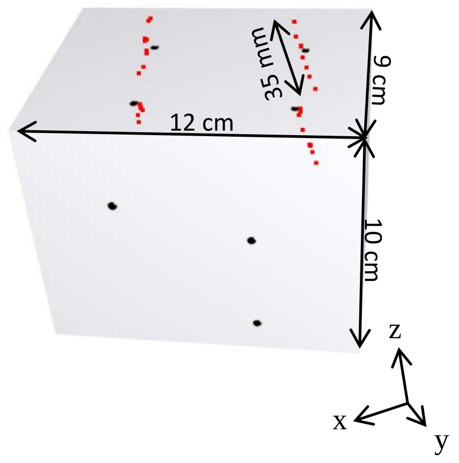


Figure 50 Front view of the point cloud.

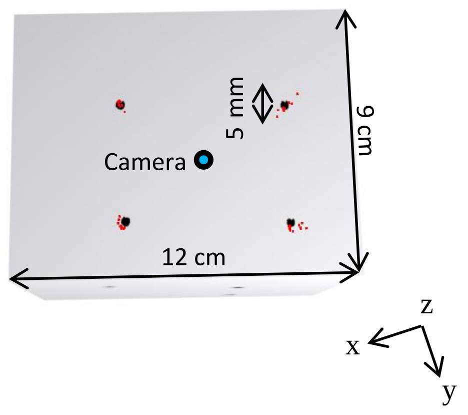


Figure 51 Top view of the point cloud.

The values in the above two projections were extracted from the generated xyz point cloud files. In this test we explicitly disabled the merging of individual points in order to see that there is a large difference in the measured depth. The error is almost 7 times larger than the in the x and y directions, because a small error in the corner point matching results in large errors in the calculated depth. E.g. the relation between measurement errors and calculated x and y location is linear where the relation for the depth is nonlinear, see also eq. 30.

6.1.3 LOCALIZATION TEST

In this test we verify the localization of the device by comparing the calculated location with the real location coordinates. In figure 52 we can see the camera placed at a distance of 45 centimeters and height of 15 centimeters from the object.

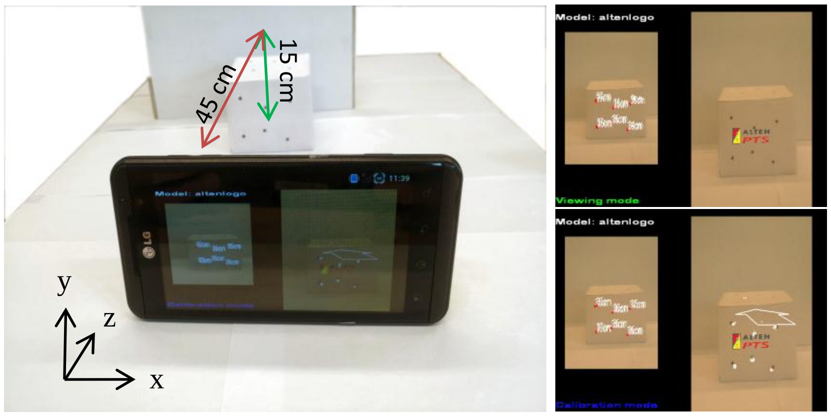


Figure 52 Localization test and screenshots from the application.

The right two windows in figure 52 visualize the augmentation of the real scene were the Alten logo is displayed in the middle of the calibration cube. The white points on the right bottom screen are world point cloud that displayed during calibration mode. We performed 91 measurements and acquired the following results.

Component	Real distance (in cm)	Mean (in cm)	SD (in cm)
X	0	0.878584408	8.936851184
Y	15	14.07093571	1.678411259
Z	45	44.50705213	2.442091036

Table 6 Results of the localization test.

The found translation from camera to world coordinates is within a centimeter of the real measured location. The large standard deviation in the x component is the result of wrong matches between the camera and world point cloud, this is probably because we have two rows of corner points which all have equal y values.

6.1.4 Corner point test

This test is performed in order to determine the performance of the corner point matching algorithm that was presented in section 4.2.1. In figure 54 we can see the world point cloud created from the corner points on the plane.

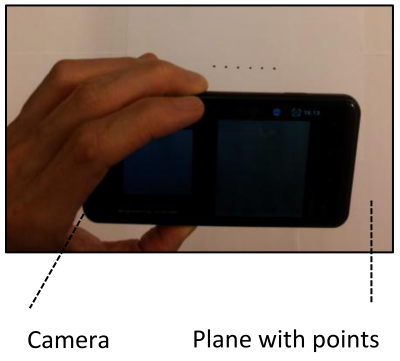


Figure 53 Camera above the plane with points in a row.

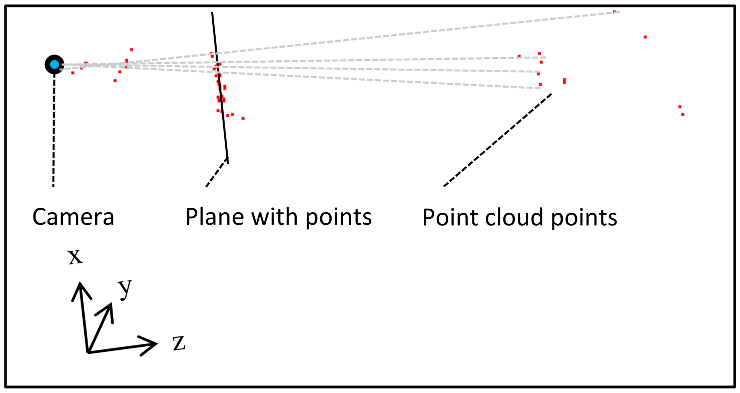


Figure 54 The point cloud created from corner points on the plane.

In total there are 61 points in the point cloud of which are wrong. This is due to the fact that the all corner points are in the same row. The corner point matching algorithm has many options to match corner points in the left camera with corner points on the right camera because they all are on the same epipolar line (see also figure 24). The high number of 24% incorrect projections does not happen in normal conditions, however in this extreme case we have many points on the same line as the epipolar line.

6.1.5 Processing test

In this test we measured the processing time spend on different tasks. In total we performed 629 measurements which are summarized in table 7.

Task	For each call (in MS)		Processing time (in %)
	Average	SD	
OpenGL Drawing	148	57	28
Creating point cloud from corner points	<1	1	< 1
Camera frame processing	332	127	37
Localization based on point cloud	304	333	29
Orientation update	<1	2	5

Table 7 Result of 629 measurements for each task.

The first column shows the specific task that was measured. In the second and third column we have the mean time and standard deviation spend on a single execution of that task. The last column shows the total percentage of time that is devoted to that task. So if a task has a low average execution time and a high processing time then it will be executed many times per second. We can see that most time is devoted to the camera frame processing, the reason for this is that frame conversions are performed in software. As can be expected, the localization takes 29% of CPU time. Based on table 7 we can calculate the number of frame per second. On average 148 milliseconds is spend on drawing a single frame. It takes 28% of the total time, thus the number of frames per second is $1.9 = (1000\text{ms} \cdot 0.28)/148\text{ms}$.

6.2 Conclusion

The research was performed in order to create augmented reality for mobile devices. The tests show that the question of creating AR without the need for markers can be answered positivity. However there are still a number of improvements that could be made.

The ability to correctly project a 3D model and thus find the translation between world an camera coordinates is dependent on the match between camera and world point clouds. Recall that the point clouds are constructed based on the found match between corner points, discussed in section 4.2.1. As we have seen in the corner point test, this matching between corner points is not always calculated correctly. And an incorrect point cloud can either result in a wrong projection of the 3D model or a low matching score and thus no projection of the 3D model. For future work more research should be performed in order to improve corner point matching without severely increasing computation time. This could for example be accomplished by introducing more constraints on the search space.

Another issue is the ability to project 3D objects over relatively large distances. The two cameras are relatively close together resulting in a more accurate depth calculation for close objects at the cost of calculated depth on objects that are further away. We can calculate the theoretically possible maximum distance that can be found by the stereo camera. Using eq. 41 for calculating depth and camera parameters in eq. 55 we have that the maximal depth would be $\mathrm{w}'' = (\mathrm{d} \cdot f_x)/(\mathrm{x}_{\mathrm{pix}} - \mathrm{x'}_{\mathrm{pix}}) = (2.5 \cdot 833)/1 = 20$ meters. However in practice an accuracy of just an single pixel is not achieved, a more likely value would be a larger difference for example of 8 pixels. It would decrease the maximum depth to only 2.6 meters. This limitation could be resolved by increasing the distance between the two cameras or using a higher resolution.

In section 6.1.5 we performed measurements in order to determine the processing time needed by the different tasks. The frame rate was calculated to be only 1.9 fps, the reason for this is that we are not able to directly access camera images in the correct format (which is an limitation of the API). Currently each frame conversion is implemented as a matrix multiplication. In OpenCV this can be performed relatively efficiently, however the operation is still in software. To overcome this problem we have to wait for an new API release.

7 FIGURES

8 BIBLIOGRAPHY

10,000 Sq. Ft. Augmented Reality Marker Sets Guinness World Record. (n.d.). (Rossi Residencial) Retrieved from http://geekbeat.tv/guinnessar/

SnapTell. (2006). Retrieved from http://www.snaptell.com/

Nokia Point & Find. (2011, April). Retrieved from http://pointandfind.nokia.com/main_publisher

Calibration. (2012). (D. Stoyanov, Producer) Retrieved 11 5, 2011, from Calibration tools: http://www.cs.ucl.ac.uk/staff/Dan.Stoyanov/calib/main.html

Google Goggles. (2012). Retrieved from http://www.google.com/mobile/goggles/#text

Air Hair: Augmented reality for those who want to learn haircutting. (n.d.). Retrieved from http://www.techfever.net/2010/06/air-hair-augmented-reality-for-those-who-want-to-learn-haircutting/

Bellman, R. E. (1997). *Introduction to Matrix Analysis* (Second ed.). New York: McGraw-Hill Book Company.

Bonsignore, E., Hansen, D. L., Toups, Z. O., Nacke, L. E., Salter, A., & Lutters, W. (2012). Mixed Reality Games. *In Proceedings of the ACM CSCW '12 Companion*, (p. 4). Seattle, WA, USA. doi:10.1145/2141512.2141517

Bouguet, J.-Y. (n.d.). *Pyramidal Implementation of TMS320DM646x.*

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV* (First ed.). 1005 Gravenstein Highway North, Sebastopol, United States of America: O'Reilly Media Inc.

Braunegg, D. J. (1989). *Location recognition using stereo vision.* Massachusetts Institute of Technology, Artificial Intelligence Laboratory. Massachusetts: MIT AI Lab.

CloudCompare. (2010). 3D point cloud and mesh processing software. Retrieved from http://www.danielgm.net/cc/

Crijns, T. (2011). *Indoor positioning systems.* Technical report, Eindhoven University of Technology, Eindhoven.

Duce, K. (2010, June 22). Bye, Bye Empty Room! . Retrieved from http://4.bp.blogspot.com/_Qf_bqu90a9Q/TB_aQqoQndI/AAAAAAAAAYs/_F_PKjO_Bf4/s1600/IMG_1755.JPG

Elcobbola. (2010, August 8). Statue of Liberty. New York City, New York, United States of America. Retrieved from http://en.wikipedia.org/wiki/File:Statue_of_Liberty_7.jpg

Hardy, D. (2011). *Augmented Reality for Landscape Development on Mobile Devices.* Master's thesis, Eindhoven University of Technology, Eindhoven.

Hua, G., Yun, F., Turk, M., Pollefeys, M., & Zhang, Z. (6 December 2011). *Introduction to the Special Issue on Mobile Vision.* Springer Science+Business Media.

Klein, G., & Murray, D. (2007). *Workspaces, Parallel Tracking and Mapping for Small AR Workspaces.* Oxford. doi:10.1109/ISMAR.2007.4538852

Kniberg, H. (2007). *Scrum and XP from the Trenches; Enterprise Software Development* (First ed.). United Kingdom: C4Media.

Kuntz, N. (2009). Template Matching. Retrieved from http://dasl.mem.drexel.edu/~noahKuntz/openCVTut6.html

Montgomery, D. C., & Runger, G. C. (2011). *Applied statistics and probability for engineers* (Fifth Revised ed.). United States of America: John Wiley and Sons Ltd.

Moussa, G., Radwan, E., & Hussain, K. (2011). *Augmented Reality Vehicle system: Left-turn maneuver study.* Assiut, Egypt: Elsevier. Retrieved August 22, 2011

Nakamoto, M., Ukimura, O., Faber, K., & Gill, I. S. (March 2012). *Current progress on augmented reality visualization in endoscopic surgery.* Institute of Technology. Massachusetts: MIT AI Lab. Retrieved from http://hdl.handle.net/1721.1/6006

Nalwa, V. S. (1993). *A Guided tour of computer vision* (alk. paper ed.). United States of America: AT&T Bell Laboratories.

Schwarz, N. (2000). Emotion, cognition, and decision making. *14*(4), 433-440. doi:10.1080/026999300402745

Shi, J., & Tomasi, C. (1994). Good Features to Track. *Computer Society Conference*, (p. 600). New York.

Simon, G., Fitzgibbon, A. W., & Zisserman, A. (2000). Markless Tracking using Planar Structures in the scene. *IEEE and ACM International Symposium*, (pp. 120 - 128). Oxford. doi:10.1109/ISAR.2000.880935

Strijbosch, L. (2006). *Statistiek - Compendium* (First ed.). Leo Strijbosch & Ventus Publishing ApS.

Toor, A. (2011, 7 31). LG Optimus 3D review. Retrieved from http://www.engadget.com/2011/07/31/lg-optimus-3d-review/

Watkins, D. S. (2002). *Fundamentals of matrix computations* (Second ed.). New York, United States of America: John Wiley & Sons.

Winn, K. (2009, 11 1). *Kellis house*. Retrieved from Blogspot: http://kellishouse.blogspot.com/2009_11_01_archive.html

Yamabe, T., & Nakajima, T. (2012). *Playful training with augmented reality games: case studies towards reality-oriented system design.* Springer Science & Business Media. doi:10.1007/s11042-011-0979-7

Zisserman, R. H. (2004). *Multiple View Geometry in computer vision* (Second ed.). Cambridge, United Kingdom: Cambridge University Press.

Appendix A

In this section all the requirements of the system can be found, they are divided into functional and non-functional requirements. The functional requirements specify what the system should do or have and the non-functional requirements describe how it should behave. For instance performance requirements are an example of non-functional requirements.

1. Functional requirements

The following requirements are functional requirements; they are grouped into 5 categories. The first category shows all the functional requirements concerning the stereo camera. The next type shows all the tracking related requirements of corner points. In the 3D objects category we define how a model should be augmented with the scene (e.g. its position shape) and also how it should be loaded. The fourth category defines all the requirements that are needed for the functionality required in orientation calculation. Finally, the last is about the peripherals of the devices (e.g. the required components of the system).

1.1. Camera

The system must be able to capture and display camera images (see user story 1).

ID	1	**Priority**	High	**Duration**	16 Hrs.
User story	As a user I want to see the images that captured from the camera.				
Motivation	The camera shows the reality which is needed in order to create AR.				
Acceptance test	Images captured from the camera are shown to the user.				

The camera of the systems must be calibrated correctly (see user story 2).

ID	2	**Priority**	High	**Duration**	16 Hrs.
User story	As a user I want to see a correct overlay of the augmented reality which is drawn on the camera image.				
Motivation	The camera should be calibrated correctly in order to create AR.				
Acceptance test	Given a fixed orientation and position in 3D it should be possible to draw a pixel on the screen that matches the 3D position.				

Given a 3D position (in centimeters) it should be drawn correctly on the camera image independent of the systems orientation (see user story 3).

ID	3	Priority	High	Duration	8 Hrs.
User story	As a user I want to see a correct overlay of the augmented reality which is drawn on the camera image that is functions independent of my orientation.				
Motivation	The orientation of the system should be used to correctly display AR.				
Acceptance test	Given an unknown orientation and position in 3D a pixel on the screen should be drawn that matches the 3D position independent of the device orientation.				

1.2. *FEATURE TRACKING*

The system must be able to track features from the camera images (see user story 4).

ID	4	Priority	Medium	Duration	24 Hrs.
User story	As a user I want the system to track points in space.				
Motivation	Feature tracking is used to recognize and track points in space, based on these and other measurements the 3D location of these points is calculated.				
Acceptance test	Individual points are tracked in space and shown on the display.				

The system must be able to derive the relative distance of a feature it is tracking (see user story 5).

ID	5	Priority	Medium	Duration	24 Hrs.
User story	As a user I want the system calculate the distance of points it is tracking.				
Motivation	If we know the distances between tracked points, we can calculate the relative location of the device w.r.t. these points.				
Acceptance test	Given a set of camera images and a tracked pixel the system can estimate the distance of that pixel and display it on the screen.				

1.3. 3D OBJECTS

The user must be able to import 3D object files (see user story 6).

ID	6	**Priority**	Low	**Duration**	8 Hrs.
User story	As a user I have to be able to import 3D models in the OBJ file format into the application.				
Motivation	The user should be able to change the augmented reality by importing external 3D models into the system.				
Acceptance test	The user must be able to load a 3D object file into the application.				

The system must be able to render 3D objects (see user story 7).

ID	7	**Priority**	Medium	**Duration**	24 Hrs.
User story	As a user I should be able to see the 3D object files.				
Motivation	The system should display the 3D OBJ models into OpenGL.				
Acceptance test	The user must be able to view 3D object files inside the application.				

The system must be able merge the 3D object with the images from the camera (see user story 8).

ID	8	**Priority**	Medium	**Duration**	8 Hrs.
User story	As a user I want to see the images from the camera merged with the 3D object model.				
Motivation	In order to create AR the system has to be able to merge the camera image with the object model.				
Acceptance test	The user must be able to see the 3D model inside the camera image.				

The user must be able to project the 3D object correctly inside the scene in order to create AR (see user stories 9, 10 and 11).

ID	9	Priority	Medium	Duration	16 Hrs.

User story	The system has to be able to project the 3D object at the correct location inside the scene.
Motivation	In order to create AR the object should be rendered at the correct location.
Acceptance test	When the camera is moved the object is projected at exactly the same location inside the scene.

ID	10	Priority	Medium	Duration	16 Hrs.

User story	The system has to be able to project the 3D object at the correct orientation inside the scene.
Motivation	To create AR object should be rotated in such a way that their orientation is correct, meaning that from a scene perspective the object orientation shouldn’t change.
Acceptance test	The object is rendered at a fixed orientation from a scene perspective.

ID	11	Priority	Medium	Duration	8 Hrs.

User story	As a user I want the system to see the 3D object at a correct size w.r.t. the scene.
Motivation	The object should have a fixed size w.r.t. the scene in order to create AR. This means that if the distance between the camera and object increases the size of the object decreases.
Acceptance test	When the camera is moved such that the distance between it and the object changes then the relative size of the object w.r.t. the scene stays the same.

1.5. Orientation

The system must be able to calculate its velocity based on the accelerometers (see user story 12).

ID	12	**Priority**	High	**Duration**	8 Hrs.
User story	As a user I want the system to measure its velocity and be able to calculate the distance of tracked objects.				
Motivation	The velocity of the system is needed in order to determine the distance of tracked objects.				
Acceptance test	When the system is moved its speed is visualized on the screen.				

The system must be able to compute its orientation using the computed pitch, roll, yaw and gyroscope values (see user stories 13, 14, 15, 16 and 17).

ID	13	**Priority**	High	**Duration**	8 Hrs.
User story	As a user I want the system to calculate the pitch based on the accelerometers.				
Motivation	Calculating the pitch using the accelerometers is needed in order to compensate for drift caused by the gyroscopes.				
Acceptance test	The pitch of the system is shown on the display.				

ID	14	**Priority**	High	**Duration**	8 Hrs.
User story	As a user I want the system to calculate the roll based on the accelerometers.				
Motivation	Calculating the roll based on the accelerometers is needed in order to compensate for drift caused by the gyroscopes.				
Acceptance test	The roll of the system is shown on the display.				

ID	15	**Priority**	High	**Duration**	2 Hrs.
User story	As a user I want the system to calculate the yaw using the digital compass.				
Motivation	Calculating yaw using the digital compass is required in order to compensate for drift caused by the gyroscopes.				
Acceptance test	The yaw of the system is shown on the display.				

ID	16	**Priority**	High	**Duration**	8 Hrs.
User story	As a user I want correct digital compass values.				
Motivation	The system should compensate the digital compass values introduced by tilt.				
Acceptance test	When the system is tilted the compass values don’t change.				

ID	17	**Priority**	High	**Duration**	16 Hrs.
User story	The system must be able to update its orientation using the gyroscopes.				
Motivation	Gyroscopes are ideally suited to calculate change in orientation.				
Acceptance test	When the system is rotated the change in orientation is visualized.				

1.7. Peripherals

The system must have the required components (see user stories 18, 19, 20, 21 and 22).

ID	18	**Priority**	High	**Duration**	1 Hr.
User story	The system has to be able to use accelerometers.				
Motivation	Accelerometers are needed in order to calculate orientation.				
Acceptance test	The device has accelerometers.				

ID	19	**Priority**	High	**Duration**	1 Hr.
User story	The system has to be able to use gyroscopes.				
Motivation	Gyroscopes are needed in order to calculate orientation.				
Acceptance test	The device has gyroscopes.				

ID	20	**Priority**	High	**Duration**	1 Hr.
User story	As a user I want the system to have a compass module.				
Motivation	The compass module is used to calculate orientation.				
Acceptance test	The device has a compass module.				

ID	21	Priority	High	Duration	1 Hr.
User story	As a user I want the system to have a screen.				
Motivation	A screen is needed in order to visualize AR.				
Acceptance test	The device has a screen.				

ID	22	Priority	High	Duration	1 Hr.
User story	As a user I want the system to have a camera.				
Motivation	A camera is needed in order to create AR.				
Acceptance test	The device has a camera.				

3. Non-functional requirements

The nonfunctional requirements specify how the system should behave. They are divided into performance and usage related requirements.

1.8. *Performance*

The system AR projection must be updated and displayed at a minimum of 1 fps (see user stories 23, 24 and 25).

ID	23	**Priority**	Medium	**Duration**	8 Hrs.
User story	As I user I want the system to perform fast enough to experience AR.				
Motivation	A slow frame rate decreases the AR experience.				
Acceptance test	The system runs at a minimum of 1 fps.				

ID	24	**Priority**	High	**Duration**	8 Hrs.
User story	OpenGL must be used to free the processor of rendering computations.				
Motivation	The use of OpenGL increases the available computation time because rendering related calculations are not performed in software.				
Acceptance test	OpenGL is implemented on the system.				

The system must be reliable in terms of operability (see user story).

ID	25	**Priority**	High	**Duration**	32 Hrs.
User story	As a user I want the system to perform reliably.				
Motivation	A system that doesn't function consistently and reliably is not desirable.				
Acceptance test	The system performs correctly during a 5 minute test.				

1.9. Usage

The system must be used in bright environments (see user story 26).

ID	26	**Priority**	High	**Duration**	0 Hrs.
User story	As a user I operate the system in bright environments.				
Motivation	This is a specific design decision of the system.				
Acceptance test	The system is used in bright environments.				

The system must either be used indoor or outside in dry and clear conditions (see user story 27).

ID	27	**Priority**	High	**Duration**	0 Hrs.
User story	As a user I operate the system in clear weather conditions.				
Motivation	The system is designed with this specific requirement in mind.				
Acceptance test	The system is used in clear and dry weather conditions.				

Appendix B

The paper on indoor positioning systems that investigates the different techniques to determine position in indoor environments.