Integration of 3D tracking systems for Interaction in Spatial Augmented Reality

Master's Thesis
Ioanna Tziouvara

December 14th 2012

Computer Graphics and Visualization Group
Department of Intelligent Systems
Faculty of EEMCS
Delft University of Technology
Integration of 3D tracking systems for Interaction in Spatial Augmented Reality

THESIS
submitted in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
in
COMPUTER SCIENCE
Media & Knowledge Engineering
by
Ioanna Tziouvara
born in Larisa, Greece

Computer Graphics and Visualization Group
Department of Intelligent Systems
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
© 2012 Ioanna Tziouvara.
The cover image displays the application of the SAR system that was implemented in this thesis, aiming to support interaction with a prototype object during rapid prototyping stages in industrial design engineering practice.
Integration of 3D tracking systems for Interaction in Spatial Augmented Reality

Abstract

In this thesis, a projector-based Spatial Augmented Reality (SAR) system designed and developed to be applied to support physical and virtual 3D Rapid Prototyping in the field of Industrial Design Engineering is presented. The main contribution is a 3D scanner to get a virtual model of a physical model and tools to support the design of features interactively, on the object’s surface.

More specifically, this work contains an approach to set the hardware to support SAR, else known as hardware calibration for SAR. Each hardware entity is calibrated with respect to a common “world”, in order to achieve effective communication. This world is set to coincide with the graphics world, and this allows us to imagine being inside the 3D graphics world while the virtual content is rendered onto the scene’s objects. In order to identify the basic ingredients that enable interaction in our SAR system, we take into account the limitations of Rapid Prototyping process, background knowledge for SAR systems and related work. Therefore, we designed the interaction components of the system according to characteristics of our setting. The SAR system was designed to perform in the peripersonal region. In this region, the user inserts input via a constructed IR tracked pen and a dynamic menu is used as interface to the system. Functionality such as selection, feedback and annotation is enabled for interacting with the SAR system. The system’s application is divided into two parts. The first part includes the use of the RGB-D camera and the IR tracker for the construction of a 3D scanner, in order to produce a virtual model of an object through sampling, segmentation, and registration of sequential point clouds. In the second part, the result of the scanning process, which is a polygonal mesh of the scanned object, is added to the SAR system’s application that enables interaction with virtual models and the ground level of the world. These two parts of the SAR system application aim to support industrial designers in the scanning of a freshly made physical prototype, and enable the design of features on the corresponding virtual model by using the SAR system during the rapid prototyping process.

In order to identify the strong points and weaknesses of the current state of the SAR system application, we carried out a user evaluation. 21 students from the Faculty of Industrial Design Engineering evaluated the two parts of the system’s application. The results show that the SAR system is useful and that it has great potential in the field of Industrial Design Engineering. Nevertheless, there is still room for improvement and future work, in order to be fully applied in the field.

Thesis committee:

Chair and Supervisor: Prof. Dr. E. Eisemann, Faculty EEMCS, TU Delft
External supervisor: Ir. J.C. Verlinden, Faculty IDE, TU Delft
Committee member: Dr. Ir. G. de Haan, Faculty EEMCS, TU Delft
Approximately a year ago, I chose to work on the field of Spatial Augmented Reality (SAR) for my master’s thesis, after identifying some of the challenges that this field offers. I was truly amazed by the potential of this field and my excitement was such that I started imagining how this technology could improve real life situations. My literature study helped me get a first taste of SAR and its progress through the years. I definitely went deeper when I started designing and implementing my system. During this project I had to deal with various kinds of challenges and take initiative to find solutions. This project involved fields such as Computer Graphics, Computer Vision, Geometry and Linear Algebra and Software Engineering. Furthermore, software installations and troubleshooting was an everyday experience.

At this point I would like to thank all the people that were involved in any kind of way in this project. First of all, my supervisors; the thesis started with Gerwin de Haan as supervisor and continued with Jouke Verlinden and Elmar Eisemann. I want to thank each one for giving me the chance to work with cutting-edge technology, for providing me with valuable advice and feedback and for contributing to my motivation and inspiration during the various stages of this thesis. Furthermore, I would like to thank the PhD candidate Christian Kehl for providing me with his opinion in various graphics-related topics and for the company during the final phase of the project.

Last but not least, I want to thank the people that are next to me all these years and support me in good and difficult times. I am forever grateful to my parents for making it possible for me to do my second master’s abroad. Moreover, I want to thank my best friend Helena Michouli for being there for me at any time. Finally, I want to thank my partner George Siokas for his patience with me during this project, for believing in me and for making me focus on the positive side of facts.

Ioanna Tziouvara
Delft, the Netherlands
December 1, 2012
# Table of Contents

**Abstract** .................................................................................................................. 5
**Preface** ....................................................................................................................... 7
**Table of Contents** ..................................................................................................... 9

**PART I - INTRODUCTION** ...................................................................................... 11

**Chapter 1: Introduction** .......................................................................................... 12
  1.1 Motivation .............................................................................................................. 12
  1.2 Problem Definition and Goals ............................................................................. 13
  1.3 Thesis Structure .................................................................................................... 14

**Chapter 2: Related Work** ......................................................................................... 16
  2.1 Background ........................................................................................................... 16
  2.2 Related Work ......................................................................................................... 17
  2.3 Summary .............................................................................................................. 24

**PART II - THE SYSTEM** .......................................................................................... 25

**Chapter 1: System Overview** .................................................................................. 26
  1.1 The Reference System ......................................................................................... 26
  1.2 Our SAR System .................................................................................................. 27

**Chapter 2: Hardware Calibration** .......................................................................... 30
  2.1 Theoretical background of Camera Calibration and Existing Work ................... 30
  2.2 Real World as Common Reference for System Calibration ................................. 47
  2.3 Tracker to World Calibration .............................................................................. 53
  2.4 Projector to World Calibration ............................................................................ 53
  2.5 RGB-D Camera to World Calibration .................................................................. 59
  2.6 Summary .............................................................................................................. 66

**Chapter 3: Interaction Design** ................................................................................. 67
  3.1 About Interaction .................................................................................................. 67
  3.2 Defining the Spatial Realm for Interaction ........................................................... 67
  3.3 User Input ............................................................................................................. 68
  3.4 User Interface ....................................................................................................... 70
  3.5 Functionalities ...................................................................................................... 72
  3.6 Summary .............................................................................................................. 74

**Chapter 4: 3D Scanner and Application** ................................................................. 75
# Chapter 4: About 3D scanner and application to RP

- About 3D scanner and application to RP: 75
- Getting the Virtual Model via Scanning: 75
- Interaction with the reconstructed Model and the World: 80
- Summary: 83

## Part III: Conclusions

- Chapter 1: Evaluation: 86
  - About Evaluation: 86
  - Performance Analysis: 86
  - User Evaluation: 92
- Chapter 2: Conclusions and Future Work: 107
  - Conclusions: 107
  - Future Work: 108
- Bibliography: 110

## Part IV: Appendices

- Appendix A: Implementation: 114
- Appendix B: Code Snippets: 115
- Appendix C: Hardware Components: 118
  - Projector: 118
  - Tracker: 123
  - RGB-D Camera: 125
- Appendix D: User Evaluation Material: 132
PART I
INTRODUCTION
Chapter 1: Introduction

This chapter introduces the reader to the topic of the present work. It starts with a description of the field of application, continues with an elaboration on the problems the field faces, and presents the goals of the thesis. The chapter ends with an explanation of the document structure.

1.1 Motivation

In order to create a fully functional prototype, industrial design engineers make use of Rapid Prototyping method. According to wikiversity\(^1\), Rapid Prototyping is a process in which the designer quickly fabricates a scale model of a physical part manually or by inserting three-dimensional computer aided design (CAD) data into machines that construct physical objects, like 3D-printers. The prototypes are usually incomplete examples of what a final product may look like. Each time a prototype is used, a formative evaluation gathers information for the next, revised prototype. This process continues to refine the product until the final needs and objectives are met. The following diagram (Figure 1.1) demonstrates the Rapid Prototyping model.

![Rapid Prototyping model diagram](http://toolboxes.flexiblelearning.net.au/demosites/series9/906/ddw_respak/ddw_e1/html/ddw_e1_plan.htm)

Rapid Prototyping usually involves a large number of iterations in order to approach the final design. The development of each intermediate prototype consists of CAD-aided modeling and physical model making. For each iteration, an updated version of the prototype needs to be prepared. Considering that this involves the use of new materials, the overall process is costly. Furthermore, according to Gibson\(^2\), Rapid Prototyping is not as quick as it might be expected. It might take hours or even days to

---

prepare the updated version of the prototype. Therefore, there is a demand for further increase in speed.

Fast-paced industry creates emerging needs for faster and cost-efficient product development. This implies the need for quicker and cheaper rapid prototyping processes. Nowadays, technology has greatly evolved, offering solutions to overcome a majority of problems. At this point, technology might have a solution for making the rapid prototyping process more efficient in terms of cost and time consumption.

Spatial Augmented Reality registers virtual information on physical surfaces. This has a direct translation to Rapid Prototyping, corresponding to the merging of CAD models with physical models. Projecting virtual content on physical surfaces has the potential to decrease the number of physical prototypes being constructed during the process, thus decrease the overall cost of the end product. The same physical model can be reused until its geometry meets the design goals of the prototype’s next version.

1.2 Problem Definition and Goals

There are two major limitations concerning the process of Rapid Prototyping (RP). The first one is that it is not as ‘rapid’ as many people realize and would prefer to be. The second one is that it is a costly procedure. We begin with the analysis of the first problematic area regarding time-consumption. As mentioned earlier, in each cycle of RP, a prototype is prepared. For that, industrial designers spend time to prepare the CAD model in order to reach an updated version, embodying all changes to the design. The more the design changes, the merrier the time needed to incorporate the changes to the CAD model. As soon as the three-dimensional CAD model is ready, it is sent to a 3D-printer in order to build a physical model of the prototype. Figure 1.2 illustrates an example of a prototype’s 3D-print. In cases where industrial designers build the physical model from different materials (i.e. clay) and without using additive manufacturing technology (3D printing), the overall speed of RP decreases dramatically. Note that for the next prototype version, they have to build yet another model.

![Figure 1.2: A functional prototype as a result of 3D-printing. The prototype consists of hardware and a white case to cover the hardware. The case was previously created by using CAD-aided modeling. A 3D-printer is then used to create its physical model. Image taken from 3ders](http://www.3ders.org/articles/20120303-give-the-raspberry-pi-an-attractive-3d-printed-case.html)
The second problem, which also happens to be indirectly linked to the first, is the cost of the overall procedure. As time equals money, the expenses increase with respect to the time spent during the RP process. In cases where extensive human labour is replaced by the use of additive manufacturing technology, the process of rapid prototyping can be seriously reduced. Nevertheless, whilst most machines are networked to assist file transfer, machine operation still requires a significant amount of manual labour, thus 3D printing is an expensive service, which results in an immediate and significant increase of the overall cost of RP process. However, even in the ideal scenario where 3D-printing costs could be discarded, another factor that increases the cost of RP is the extensive use of materials for each physical model. As the number of RP cycles increase, new materials are needed for each fresh version of the prototype.

The goal of the present thesis is to approach the afore-mentioned problems and provide a solution in order to reduce cost and speed up the process of RP. As mentioned earlier, the field of Spatial Augmented Reality offers a lot of possibilities towards the establishment of a new system to support Rapid Prototyping, aiming to limit the problems. More specifically, the use of projection technology allows the projection of virtual information onto physical surfaces of human environment. Similarly, virtual content (3D CAD model) can be projected on a material surface (physical model). This offers the potential of reusing the same physical model, since the model stays intact (the content is virtual). Furthermore, the development tools, supporting basic operations during Rapid Prototyping -such as design of prototype’s features-, would enable the interactive design of prototype’s components. Thus, the overall 3D modeling time can be significantly reduced.

As already mentioned in Section 1.1, continuous evaluation of each prototype is crucial for the design development. The same applies to the present work as well. The first step would be to gather all the necessary information and directly input this knowledge to the design of the proposed/developed SAR system. Additionally, user evaluation of the SAR system is necessary, in order to validate the developed system. For the best possible realistic results, industrial designers are chosen for this SAR system’s evaluation.

To sum up, in a few words, the main goals of this thesis are the following:

- Design and development of a Spatial Augmented Reality System for Rapid Prototyping
- Evaluation of the developed SAR system by its actual end-users (industrial designers).

1.3 Thesis Structure

This thesis report is divided into three main parts. The current part -Part I-, apart from the introduction, problem definition and goals, provides information and background knowledge on the field of Spatial Augmented Reality, including relevant and inspiring work for this thesis.

Part II is dedicated to the analysis of the developed SAR system. After providing an overview of the system in the first chapter, we move on with the detailed description of the approach followed for the calibration of the system’s hardware components (Chapter 2), in order to achieve a basic functional system. The third chapter of Part II includes a detailed analysis of the system’s interaction design, as well as details on the development of the components to support interaction with the SAR system. The
last chapter of Part II (Chapter 4) presents in detail the 3D scanner application and the enabled support for rapid prototyping application.

The third part of this report -Part III- consists of two chapters. Chapter 1 contains the evaluation of the SAR system, while the second one (Chapter 2) consists of the conclusions and suggestions for future work.

Last but not least, Part IV contains appendices, necessary to get a comprehensive impression of the overall work.
Chapter 2: Related Work

In this chapter, background on the field of Spatial Augmented Reality is presented and work related to it is being discussed. More specifically, this chapter starts with more general information about SAR, such as its pros and cons and then moves towards the description of related work with respect to Interaction, Scene, Technology and Application, sections which form a SAR system. The chapter ends with a short summary.

2.1 Background

Spatial Augmented Reality (SAR) is a branch of the well-known research field of Augmented Reality (AR). It was firstly introduced by Raskar, Welch and Fuchs as Spatially Augmented Reality in 1998 [Raskar 98] and is still a fast-growing field, considering the number of publications per year. Being a category of the AR field, SAR adheres to Azuma’s definition for Augmented Reality [Azuma 97]: it combines real and virtual imagery, it is interactive in real-time and registers the virtual imagery to the real world. The substantial difference between spatial and conventional augmented reality lies in the approach that is followed for displaying the virtual imagery in each case.

Spatial augmented reality makes use of projection technology and decouples the display surface from the display device. Digital projectors -which are the display devices-, are used to facilitate the display of computer-generated (virtual) imagery on physical objects or surfaces, which are the display surfaces. The decoupling of display devices and surfaces has the potential to provide a more natural way of interaction, as the virtual information registered to physical objects is directly integrated in user’s environment, i.e. the real world.

Most of the existing research focuses on projecting onto planar surfaces. For relevant examples, refer to Beardsley et al. [05], Cao et al. [06] and Ehnes et al. [04]. Nevertheless, according to Raskar et al. [01], the augmentation does not have to be limited to 2D flat surfaces. The use of a projector as a display device affords the possibility of illuminating physical objects with complex 3D geometry. The projection onto complex physical surfaces sets the content free from the confines of a limited flat monitor display [Jones 10]. This can revolutionize the way interfaces are designed till now.

Advantages

According to several research works like Raskar et al. [98], Zhou et al. [08], and Mitasova et al. [06], there are plenty of benefits offered by SAR. First of all, the user does not have to wear cumbersome equipment, like head-mounted displays, therefore SAR is known for its minimal intrusion. Moreover, the eye accommodation is easier, since virtual content is rendered nearby the real world location, resulting in motion sickness elimination. Furthermore, the augmentation can be visible to several users, therefore it supports collaborative scenarios. Projection technology provides a large field of view, higher resolution and bright images of virtual objects. A key feature of SAR is the integration in almost any common working environment. The merging of the virtual and the physical worlds can provide a unique immersive experience. From that perspective, tangible experience with interactions
that are not possible in a truly physical environment can also be enabled. Finally, the combination of multi-projectors provides a much larger area, undistorted projection, multiple viewing angles, and alleviation of self-occlusion problems.

**In Future**

The realization of the SAR concept becomes increasingly feasible as projectors decrease in size, weight, cost and power consumption, according to Beardsley et al. [05], Raskar et al. [01] and Jones et al. [10]. The trend towards the miniaturization of projectors and companies’ test cases of smartphones with embedded projectors (such as Samsung\(^4\)), indicate that in the near future, pico-projectors will become a standard feature of smartphones, which will in turn have the ability to project on almost every surface, according to Cao et al. [06]. This is expected to contribute to the spread of SAR to achieve applications that were not possible before.

**Weaknesses**

Projection involving only passive viewing is of limited use, therefore there is a need for an interaction mechanism, according to Beardsley et al. [05]. Although the opportunities that accompany SAR are great, effort towards the interaction mechanisms is necessary [Mitasova 06]. This limitation is known in the research community and according to Elepfandt et al. [11], from a user-centered perspective, we still do not know how to interact with SAR environments properly. Having a human interface to a physical model is the essence of ‘intuitive’. Therefore, the goal of projector-based augmentation systems should be to make users enjoy the combined advantages of natural human interaction and computer-based user interfaces [Raskar 01].

**2.2 Related Work**

In order to better analyze related work and respective attempts towards functional SAR systems, we divide the field into four building components: *Interaction, Scene, Technology, and Application*. A small description of each follows in this section and an illustration is provided in Figure 2.1.

---

\(^4\) http://www.engadget.com/2010/02/14/samsung-i8520-halo-with-3-7-inch-super-amoled/
A user, in order to communicate with a system, has to perform specific actions. For this, special support needs to be designed and implemented. The support for interaction basically involves the means to provide user input and an interface with supported functionality to the system. Moving on to the next component, the scene is linked to the environment of the SAR setting. All the objects that have a specific role in the setting belong to the scene. For example, the object, on which the virtual content is projected, is a part of the scene. Next, the technology used to create a SAR system is being discussed in the corresponding component. Technology, such as display devices, tracking and registration techniques, are included in this section. Finally, SAR is strongly connected to the application field that it serves. Therefore, the application component provides an overview of applications where the SAR field has been applied.

### 2.2.1 Interaction

Since SAR is applied in the real world, it instantly inherits some natural aspects of human interaction (NHI). Nevertheless, SAR is still a technology, and this fact attributes to the insertion of the machine entity into the interaction. This means that in a SAR setup, we still have to deal with human-computer interaction (HCI), but since the real world plays an important role, the interaction must be more abstract, and embed features of natural human interaction. Figure 2.2 provides a visual representation of the nature of interaction in SAR, according to our opinion.

![Interaction in SAR: involves characteristics from both HCI and NHI.](image)

As mentioned previously, the term interaction refers to the user’s action in order to communicate with the system. This means that the user is the one to interact with the SAR system, therefore the user is an entity for whom the support is built for. Characteristics such as mobility and number of users play a role in interaction. For example, a user that stays at the same position (static) and a user that moves around in the environment (dynamic) will probably use different means to interact with the system. Moreover, the user has to provide input to the system, which can be obtained, either from a single, or multiple sources. The input is mainly divided into two categories: action via user’s body parts and action via extra tools. In both categories, the input can be basically anything. Nevertheless, the most common input in the first category, is input provided by tracking user’s head, hand(s), finger(s) and speech. In the second category, the most common is a tracked device such as a pen. Of course, any other artifact falls in this category. Apart from user input, it is common that the user needs additional support to interface the SAR system. A menu with options is widely used as user interface to a SAR system.
From this point and on, we go through the literature and discuss the most common features that are used in order to achieve interaction with a SAR system. Some of the works use a conventional/standard mouse, in order to provide user input. This form of input is common in classical human computer interaction. Tonn et al. [08] used a mouse and a laser pointer to interact with a SAR system that facilitates interactive architectural scenarios. In the same work, the user accesses the system via a Graphical User Interface (GUI) running on the same computer as the SAR application. The same applies for Han et al. [06], who extend the interface of standard desktop applications into physical environments (Figure 2.3a). Similar to the input via mouse, a game controller has also been reported, enabling the user to interact with the application in SAR for industrial Computer-Numerical-Control machines [Olwal 08].

Moving a step forward, the attachment of a marker to a finger tip detaches the user input from the confines of the conventional mouse. Accordingly, Porter et al. [10] tracked the finger tip of a user to provide input to their SAR system, applied to rapid prototyping of a car console (Figure 2.3b). In the same sense, retro-reflective stickers, which can be attached to any object, have the power to transform an object into an input device itself, according to Verlinden et al. [03] and Dorfmuller [99]. Chan et al. [11] developed a 3D user interface - the “MagicPad” - which is a planar small surface (pad), tracked in space, on which virtual imagery is projected and offers properties such as scaling and rotation. Similarly, Ehnes et al. [04] project options to a planar surface, in order for the user to interface the system (Figure 2.3c). They mention that the user input is provided by a tracked infrared pen with a LED on its tip, nevertheless they have not implemented it in this work. It seems that the tracked pen is a typical solution reported in several research works like [Cao 07], [Bandyopadhyay 01], [Jones 10] and [Smith 11]. More specifically, Jones et al. [10] built a Surface Interaction Engine to decouple content creation from the display surface, in order to improve the interaction with everyday objects. An infrared-emitting pen is used to interact with projected virtual content and a menu is projected to provide options onto a specified by the user location (Figure 2.3d).

The literature also reports the use of handheld projectors for user input. Beardsley et al. [05] factored out the projection motion to create a projection that is static on the display surface and they left part of the projection (a cursor at the center of projector image plane) to follow projector’s movement. The user sees a static projection, with a cursor moving across it, in response to pointing motion of projector as input (Figure 2.4a). Lately, an increasing trend towards body part(s) motion tracking has appeared. In “LightSpace”, they track the user’s hand and encode gestures such as ‘drag and drop’ for interacting with virtual content inside a room [Wilson 10]. In “OmniTouch”, the user’s finger is tracked and provides input for a number of applications developed to test the concept [Harrison 11]. See Figure 2.4b and Figure 2.4c respectively.
Finally, Mitasova et al. [06] present a SAR system for real-time landscape model interaction, which uses a tangible geospatial modeling environment. The result of the manipulation of the deformable surface by the user(s) is the input for the system. Figure 2.5 shows how the user input is inserted. This is a tangible user interface to a SAR system. A final remark here is that the interaction devices depend strongly on the tracking techniques that are used in each SAR setup.

**Touchless Multimodal interaction in SAR**

At this point, it is important to mention an interesting publication from Elepfandt et al. [11] about appropriate interaction in Spatial Augmented Reality environments. Elepfandt et al. [11] identify the lack of appropriate interaction techniques in SAR applications through a short research in literature, and propose that the multimodal and touchless interaction is the future for SAR. For touchless interaction, they investigated Gaze, Speech and Gesture, and they propose a combination of them, since each one contains certain advantages and disadvantages. In order to reinforce the suggestion for multimodality, they underline that in human-human communication, speech and gaze are normally used for complementary information, inferring that human interactions are multimodal by nature. An interesting part of their paper is the human interaction in 3D space. The spatial aspects of perception and interaction are introduced, and it is proposed to be taken into account for improving interaction in SAR. They adapt Previc’s theoretical model of 3D spatial interaction (Figure 2.6), and describe that the peripersonal region is within hand reach, while the three extrapersonal regions are out of reach. Hence, one fact is that each region depends on the individual’s arm length. The peripersonal behavioral system is characterized by grasping, reaching and manipulating objects, whereas the extrapersonal behavioral systems are characterized by the visual search and recognition of objects (focal), navigation and target orientation (action) and spatial orientation, postural control and locomotion (ambient).
2.2.2 Scene

The scene might consist of a single or multiple objects. It can be either static or dynamic. In a static scene, the objects are fixed and remain in the same position. On the other hand, the objects can “change” in a dynamic scene. This change refers to two different aspects: the first aspect is that the position and orientation (pose) of the object(s) change, and the second aspect is the fact that an object deforms, meaning that the shape of the object changes.

Many different examples of scenes can be found in literature. At this point, we refer to the most representative ones. Tonn et al. [08] used the walls of a room as a surface to project on, since their application involved in-situ comparison of architectural plans (Figure 2.7a). The scene in this example consists of the walls on which the virtual content is registered. The scene is apparently static. Similar scenes appear in works such as [Han 06], [Forlines 05] and [Cao 07]. On the other hand, Jones et al. [10] construct the scene of the system on the fly. This means that the system is designed to operate with a variety of scenes. As Figure 2.7b shows, they build a scene out of styrofoam the first time and a scene out of sand the second time. Similarly to the previous works, the scene is static during the SAR system’s operation. Bandyopadhyay et al. [01] use a house model as part of the scene which is being
tracked, thus it is dynamic (pose changes). Furthermore, the menu used to interface the system, is also a part of the scene (Figure 2.7c).

![Figure 2.7: Scene setups: a) Use of the user’s body as part of the scene [Wilson 10], b) Use of hand as part of the scene [Harrison 11].]

Apart from rigid objects, which can be either static or dynamic in a scene, the user’s body can be encoded to act as a part of the scene too. In “LightSpace” system, the body is used as a surface in order to enable the connection of two other surfaces (Figure 2.8a). Therefore, Wilson et al. [10] use the body as part of the scene. The body is tracked as an entire volume (dynamic). The same applies to “OmniTouch”, where one of their application examples is the use of the palm of the hand as an active display surface (Figure 2.8b). In the same sense, the hand becomes a part of the scene, which is dynamic and deformable. Another interesting example of a deformable object which is part of the SAR system’s scene, is the deformable surface used as a tangible geospatial modeling surface by Mitasova et al. [06] (Figure 2.5). Refer to Maas et al. [11] for a deformable material created especially for use in SAR.

### 2.2.3 Technology

Tracking and Registration, and Display Technology are the fundamental components of the technology being used in spatial augmented reality. The main goal in SAR is to augment one or more objects of the real world with virtual imagery. To do so, the 3D position and orientation (pose) of the real object is required. Most cases deal with moving objects. This means that the pose is needed to be available in real-time. For that, real-time tracking of the object is necessary. The augmentation of the object into the real world takes place by registering the virtual information to the tracked object. If the position and orientation is known, then the virtual information can be precisely overlaid onto the object.

**Tracking and Registration:** Several tracking techniques exist and have been deployed to achieve tracking of real world objects in augmented reality applications. Tracking research focuses mostly on sensor-based and vision-based, according to Zhou et al. [08]. Sensor-based tracking techniques are based on sensors such as magnetic, acoustic, inertial, optical, and mechanical. Each category has certain advantages and disadvantages. In most cases, the choice of a specific sensor-based technology for tracking relies on application’s requirements. For example, ambient light and infrared radiation affect optical tracker’s performance, so in such cases, optical tracking is not the best option. The survey for tracking technologies by Rolland et al. [01] provides more details. Sensor-based tracking is
very common in SAR systems. Chan et al. [11] use an optical sensor in order to track the pad and the pen. Similarly, Bandyopadhyay et al. [01] track the pen and the house object by using magnetic tracking. On the other hand, vision-based tracking is an active research area, according to Zhou et al. [08], and engages computer vision methods for achieving tracking. Porter et al. [10] use computer vision to detect the color of the marker and track its position. Furthermore, feature-based tracking and model-based tracking are quite common methods. The first searches for a correspondence between the 2D image features and their 3D world frame coordinates (i.e. fiducial markers), whereas the second uses a model of features of tracked objects or a 2D template of the object based on distinguishable features. Ehnes et al. [04] attached a fiducial marker on the menu surface to track it in space. Moreover, the introduction of RGB-D camera by Microsoft -else known as Kinect- resulted in a new form of tracking which is based on 3D Image Processing. Wilson et al. [10] use Kinect to track volumes in “LightSpace”, and Harrison et al. [11] achieved finger tracking with Kinect in “OmniTouch”. Similar to Kinect, depth sensing technology, such as 3D laser scanners, are also used to track changes in the environment. For an example, refer to [Mitasova 06].

**Display Technology:** In conventional augmented reality we refer to head-mounted displays, desktop monitors, or handheld displays such as cell phone screens, as the enabling technology for display [Carmigniani 10]. One factor that differentiates spatial augmented reality from traditional augmented reality is, in fact, the display technology. SAR’s basic concept to render virtual objects directly within or on the user’s physical space, makes the physical surface the display. This is achieved by digital light projection technology. Digital light projectors serve as the display devices, whereas the physical surfaces as the display surfaces [Raskar 01]. There are two basic types of projection technology based on the usage scenario; standard projectors and pico-projectors. Standard projectors are mostly used in fixed settings, i.e. mounted to the ceiling, like in Dynamic Shader Lamps [Bandyopadhyay 01], but can also be found to be used as handheld devices like in Zoom-and-Pick [Forlines 05] and Interactive Projections [Beardsley 05], since most of today’s projectors are compact enough for handheld use [Beardsley 05]. Pico-projectors, or else pocket projectors, have the size and the weight of a cell phone and are handheld devices by origin, see [Harrison 11]. A moving projector can either be handheld or movable. Ehnes et al. [04] worked on a SAR system with a movable (rotatable) projector. Furthermore, depending on the setting, a single or multiple projectors might be used in a SAR system. Regarding display surfaces, the ideal surface is a light colored diffuse object with smooth geometry. According to Raskar et al. [98], it is practically impossible to render vivid images on highly specular, low reflective, dark surfaces.

2.2.4 Application

SAR systems depend strongly on the application they are designed to serve. SAR systems are reported to be used in a variety of fields. Tonn et al. [08] developed a system that integrates architectural software with SAR-enabling software to facilitate interactive SAR architectural scenarios. Han et al. [06] made use of projection technology to extend the interface of standard desktop applications into physical environments, aiming at office related interaction. Furthermore, Olwal et al. [08] integrated the industrial process data with the workspace behind the safety glass, in order to simplify machine operation, by merging the process data with the process itself, so that the operator’s attention stays at one place. For improving the quality and speed of the iterative process of appliance design, Itzstein et al. [11] used SAR. Similarly, Porter et al. [10] and Verlinden et al. [03] validated SAR for interactive rapid prototyping. Moreover, a use case of the work of Bandyopadhyay et al. [01] in Dynamic Shader
Lamps was the drawing application. The application was informally tested by a kid. This scenario fits educational purposes as well. Jones et al. [10] demonstrated three examples of applications, namely a golf game, a tanks game and photo viewer. The link to entertainment is apparent. Wilson et al. [10] focused on exploring a variety of interactions and strategies related to interactive displays and the space they inhabit, which can easily be applied for presentation purposes. Finally, Mitasova et al. [06] worked on real-time landscape model interaction using a tangible geospatial modeling environment. They aimed to achieve a more intuitive collaborative interaction with digital landscape data. This might be useful for educational purposes, simulations and geological studies.

2.3 Summary

In this chapter, which is the second introductory chapter of Part I, the background of SAR has been provided with respect to its definition, advantages, weaknesses, and vision. Furthermore, we took the initiative to define four building components - Interaction, Scene, Technology and Application - for SAR, in order to be able to describe this complex field. Finally, characteristic related work was presented for each building component, in order to get an overview of the solutions that have been used in SAR so far.
PART II
THE SYSTEM
Chapter 1: System Overview

In this chapter the overview of the system that was designed and developed to support Rapid Prototyping in Industrial Design Engineering is provided. The first part of the document (Part I) revealed the background information, based on which, we now define the structure of our system in terms of Interaction, Scene, Technology and Application.

1.1 The Reference System

Verlinden et al. [03] designed and implemented a SAR system for enabling rapid prototyping in industrial design engineering. More specifically, a graphical user interface for providing design options was developed and the user input was provided via a mouse. Furthermore, the scene consisted of a turntable holding the 3D-print of a car model and a static menu for options. Finally, the system’s hardware components were an LCD projector as a display device and an infrared optical tracker for tracking and registration. Figure 1.1 demonstrates Verlinden’s SAR system.

![Figure 1.1: The system: a 3D model of the car is projected on the physical model of the car. Input in inserted by mouse via a menu projected on a fixed location.](image)

By going deeper into the system’s application and from Figure 1.1, a number of system design characteristics emerge:

- The user can have multiple views of the projected virtual model on the physical one, by rotating the turntable.
- The graphics are rendered real-time and follow the movement of the tracked physical model.
The user changes the virtual model characteristics with the help of a graphical user interface projected on a specific location.

- The user input is accomplished via mouse.
- A projector and a tracking device are used as hardware components to produce this result.

The system of Verlinden et al. [03] is used as the main reference for our own system. This SAR system demonstrates a number of points which we considered to incorporate in our system. More specifically, we favor the idea of projecting on the tracked physical object lying on the turntable, since in this way, the presented virtual information is indeed interactive and, at the same time comprehensive. Furthermore, the idea of a graphical user interface for enabling rapid prototyping tasks seems appropriate. Users such as industrial designers are used to interacting with virtual models in CAD programs, thus a GUI that offers options like coloring and texturing, could support relevant tasks, such as adding features on a model that was created during rapid prototyping.

On the other hand, according to our opinion, there is still room for refinement. First of all, the fact that the menu is static at a certain position does not provide flexibility for use in settings of different layout. Moreover, the user input via mouse is not the ideal solution, taking into account that the user will be handling this projected menu by standing in front of a computer to insert the input.

1.2 Our SAR System

The purpose of the current thesis is the design of a system that supports industrial designers during the Rapid Prototyping (RP) process, aiming to increase the speed of the process and decrease the cost long term. The field of Spatial Augmented Reality has the potential to provide support for the design and the development of a system for rapid prototyping. The system presented in the previous section is a proof of this concept and a promising start.

Our vision involves the creation of a system that supports design of features on to the physical prototype in an interactive manner. The system developed by Verlinden et al. [03], carries out the changes by selecting specific options from the menu, which results to a change in the displayed virtual model. For us, the direct interaction with the physical prototype is of great importance, since we believe that this enables the designer to manipulate the prototype intuitively, as if designing features on an object with a real pencil.

Moreover, one important remark here is that during RP, the physical prototype can either be a result from a CAD model sent for 3D-printing, or manually manufactured object, by using a malleable material, such as clay. In cases where the virtual model of the prototype is created before the physical prototype, like in Verlinden et al. [03], then, the focus shifts on creating the appropriate support tools to manipulate the virtual model, while it is being projected on the physical object itself. On the other hand, when the design development begins from a physical prototype, there isn’t any immediate correspondence to its virtual one. Therefore, it would be useful if this reversed possibility was also offered. The scanning of the object is required for achieving such a result.
The figure above (Figure 1.2) shows the skeleton of a SAR system. In the diagram, the user inserts input through some means that enable interaction, resulting in a change of state. This change takes place on the scene and the effect is recorded by the tracking system. After some processing the corresponding information is then projected back in the scene. Based on the aforementioned and the building components forming the SAR system displayed in Figure 2.1 of Part I, we continue with an elaboration on the system’s basic components (interaction, scene, technology and application).

**Interaction**
The system is designed for a single user, but it is easily extensible to support multiple users. The user is considered static, since he/she isn’t being directly tracked. Nevertheless, the user input is provided by a tracked pen, and one could claim that the user is being indirectly tracked. An interface, in form of a spatial menu, is provided to the user, in order to support design tasks during RP. We refer to the menu as spatial due to the requirement of mobility in space. Finally, the major functionality offered by the system, is the Annotation functionality. Annotation on a physical object equals the design of virtual content on this object. In Figure 1.2, the user and the interaction are represented by green and blue colors respectively.

**Scene**
The scene components consist of the physical prototype(s), the physical surface of the dynamic menu and surfaces encoded as active for specific purposes (e.g. the table top surface is used as a bench for annotation, see details in Chapter 4). Apart from the menu, the physical prototype(s) will also be tracked (dynamic pose), in order to provide a more intuitive feeling during the design process. Finally, a turntable (circular surface) that enables rotation around the axis perpendicular to its own surface (to which we will refer from now on as cheese platter), is used only during the scanning application of our system. This surface is also being tracked (dynamic) and is considered as part of the scene apparently.
**Technology**
In order to enable the display of virtual content directly on the scene, a digital projector is used. For the tracking of physical objects, an IR optical tracker is incorporated to the hardware components. Note that the retro-reflective stickers detected from these devices, have the power to transform any object to an interactive device. Finally, for the purposes of the scanning of the object—in order to get its virtual model—, we fused two 3D tracking systems. The first one is the aforementioned IR optical tracker, and the second one is an RGB-D camera. Appendix C provides more information about the hardware components.

**Application**
As already mentioned in previous sections, the system aims to support Rapid Prototyping process in Industrial Design Engineering.

The following figure (Figure 1.3) provides a visual representation of our system. More specifically, the interaction components (pen and menu), part of the scene (cheese platter, cube, pen, and menu) and the technology (projector, IR optical tracker and RGB-D camera) are illustrated.

![Figure 1.3: Our SAR system in terms of technology, scene and interaction.](image)

The following chapters of this part go deeper into the developed SAR system. Chapter 2 provides all the necessary details on setting up all hardware with a common reference, in order to provide concise information. Chapter 3 is dedicated to the interaction design and its implementation, while Chapter 4 elaborates on both the 3D scanner and the system application, developed to assist the user in performing basic tasks related to Rapid Prototyping.
Chapter 2: Hardware Calibration

This chapter contains the calibration of each hardware component with respect to a common reference, in order to provide concise information. Furthermore, the chapter starts with the presentation of camera calibration background, continues with existing work, and finishes after providing all the details of the approach that was eventually followed.

2.1 Theoretical background of Camera Calibration and Existing Work

In this section, the theoretical background of camera calibration based on the pinhole camera model is described. We decided to provide all the steps, in order to gather all the necessary information and background needed for camera calibration in one section. Then, we provide information about four specific software packages that we tried for calibrating our hardware components, as well as the reasons we had to do our own calibration.

2.1.1 The Pinhole Camera Model

A pinhole camera is a simple optical device in the shape of a closed box. In one of its sides is a small hole - which is considered as a point - which is used as aperture. Light from a scene passes through this single point and projects an inverted image on the opposite side of the box, as shown in Figure 2.1.

![Figure 2.1: Principle of a pinhole camera: light rays from an object pass through a small hole to form an inverted image, taken from wikipedia](http://en.wikipedia.org/wiki/Pinhole_camera)

The pinhole camera model refers to the mathematical relationship between a 3D point of a scene and its 2D projection on the pinhole’s camera image plane. The model does not take into account lens distortions and blurring effects. The pinhole camera model is a first order (linear) approximation of the mapping from a 3D scene to a 2D image, and its validity depends mostly on the quality of the
camera. For example, the validity of the model decreases while moving from the center of the image to the edges, as lens distortion effects usually increase towards this direction. Nevertheless, the pinhole camera model is considered a reasonable depiction of a 3D scene by a camera, and that is the reason of its common use in computer vision.

The geometry of the pinhole camera model is illustrated in Figure 2.2, and is explained shortly in the following bullet points, according to wikipedia⁶.

- First of all, we consider a 3-dimensional orthogonal coordinate system with its origin at O. Note that the origin coincides with the location of the camera aperture. The three axes of the coordinate system are named \( X_1, X_2, \) and \( X_3 \), respectively. The \( X_3 \) axis is pointing in the viewing direction of the camera, which is also known as the optical axis or principal axis. The image plane is parallel to \( X_1 - X_2 \) axes, and its distance from the origin \( O \) is equal to \( f \) (known as focal length), in the negative direction of \( X_3 \). At a point \( R \), which is known as the principal point or image center, the optical axis intersects with the image plane.

- The 3D scene (world) is projected through the camera aperture (origin) onto the image plane. If \( P \) is a 3D point in the world at \( (x_1, x_2, x_3) \) coordinates relative to \( X_1, X_2, \) and \( X_3 \) axes, then the projection of \( P \) onto the image plane will result to point \( Q \). The green line that connects point \( P \) and \( Q \) and passes through the origin \( O \) (aperture) is the projection line. The origin in this case is known as the center of projection or projection center. There is a 2D coordinate system in the image plane with origin at \( R \) and with axes \( Y_1 \) and \( Y_2 \) which are parallel to \( X_1 \) and \( X_2 \), respectively. The relative coordinates to \( Y_1 \) and \( Y_2 \) axes of \( Q \) point are \( (y_1, y_2) \).

With the use of similar triangles, the mapping from 3D to 2D can be easily explained. Figure 2.3 is similar to Figure 2.2 with the difference of looking down in the negative direction of the \( X_2 \) axis. In

---

this figure, two similar triangles are observed, both having parts of the projection line as hypotenuses. The left triangle has \(-y_1\) and \(f\) as catheti (perpendicular), whereas the right one has \(x_i\) and \(x_3\).

Since these triangles are similar, we have:

\[
\frac{-y_1}{f} = \frac{x_i}{x_3} \quad \text{or} \quad y_1 = -f \frac{x_i}{x_3} \quad (\text{Eq. 2.1})
\]

Similarly, if we use Figure 2.3, but looking down in the negative direction of the \(X_1\) axis, again, two similar triangles are observed. The left triangle has \(-y_2\) and \(f\) as catheti, whereas the right one has \(x_2\) and \(x_3\). And again, since these triangles are similar, we have:

\[
\frac{-y_2}{f} = \frac{x_2}{x_3} \quad \text{or} \quad y_2 = -f \frac{x_2}{x_3} \quad (\text{Eq. 2.2})
\]

The previous two equations can be summarized in:

\[
\begin{bmatrix}
  y_1 \\
  y_2
\end{bmatrix} = \begin{bmatrix}
  -f & x_i \\
  f & x_3
\end{bmatrix} \begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix} \quad (\text{Eq. 2.3})
\]

The mapping from 3D to 2D coordinates described by a pinhole camera is a perspective projection followed by a 180° rotation in the image plane. This corresponds to the way a real pinhole camera operates. Nevertheless, we expect that a conventional camera would provide us with non-rotated images, therefore, we would like to tune the previous equations to give a non-rotated image. The simplest way to achieve this is to create a virtual front image plane, instead of the one described above (see Figure 2.2), that will intersect the \(X_3\) axis at \(f\) instead of at \(-f\) and rewrite the previous equations. The resulting mapping from 3D coordinates to 2D image coordinates is now given by:

\[
\begin{bmatrix}
  y_1 \\
  y_2
\end{bmatrix} = \begin{bmatrix}
  f & x_i \\
  x_3 & x_2
\end{bmatrix} \begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix} \quad (\text{Eq. 2.4})
\]
2.1.2 Camera Calibration adhering to Pinhole Model

After having provided the basics concerning the pinhole camera model in the previous section, we continue with the mechanics for camera calibration based on the discussed camera model. A camera that can be described by the pinhole camera model consists of two types of parameters, namely the Intrinsic and Extrinsic Camera Parameters. These parameters need to be recovered in order to be able to reconstruct the 3D structure of a scene from the pixel coordinates of its image points.

The present the section starts with a description of the two types of the camera parameters, then continues with a step by step presentation and explanation of the mathematical background that is needed in order to decompose the calibration matrix and comprehend the role of each parameter in the camera model.

Camera Parameters

The extrinsic -or else external- parameters of a camera define the position and orientation of the camera coordinate system with respect to a known world coordinate system. Therefore, we suppose that the camera does not have its center of projection at (0,0,0) and that is oriented in an arbitrary fashion. The translation vector \((T)\) between the relative positions of the origins of the two reference frames (world and camera) and the rotation matrix \((R)\) that brings the corresponding axes of the two frames into alignment (i.e., onto each other) are considered extrinsic camera parameters.

The intrinsic -or else internal- parameters of a camera link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame. The principal point -or else image centre- \((x_0, y_0)\), the focal length \((f)\) and the distortion coefficients \((\tau)\) are considered intrinsic camera parameters. The intrinsic parameters of a camera can also be obtained via the manufacturer and usually stay constant.

A. Central Projection

Consider a camera model according to the pinhole camera model. The camera coordinate system is a Cartesian coordinate system having \(x, y, z\) as main axes which are orthogonal to one another. The origin of this coordinate system is located at a 3D point \(C\) which is known as center of projection (see bullet points in the previous section). As mentioned in Section 2.1.1, the pinhole camera model is based on the projection of a point from the camera coordinates into the image plane. Figure 2.4 provides a visual representation of the central projection resulting from the pinhole model.

Figure 2.4: Central projection according to the Pinhole camera model, taken from Ramani Duraiswami’s cmsc828d notes.

7 http://www.umiacs.umd.edu/~ramani/cmsc828d/
Consider a point in the scene (labeled as scene point in Figure 2.4) represented by \( x_s, y_s \) and \( z_s \) coordinates. The distance between the center of projection and the image plane is represented by the focal length \( f \). The projection from 3D camera coordinates \((x_s, y_s, z_s)\) into the 2D image coordinates \((x_i, y_i)\) through the center of projection point \( C \), is carried out via using similar triangles, as described in Section 2.1.1. This results in the following equations:

\[
\begin{align*}
x_i &= f \frac{x_s}{z_s} \\
y_i &= f \frac{y_s}{z_s}
\end{align*}
\]

(Eq. 2.5)

(Eq. 2.6)

The previous two are summarized in:

\[
(x_i, y_i) = (f \frac{x_s}{z_s}, f \frac{y_s}{z_s})
\]

(Eq. 2.7)

This 2D image point is rewritten in homogeneous coordinates:

\[
(x_i, y_i, 1) = (f \frac{x_s}{z_s}, f \frac{y_s}{z_s}, 1)
\]

(Eq. 2.8)

This equivalence allows us to write the projection from a 3D point \((x_s, y_s, z_s)\) in camera coordinates to its 2D image point \((x_i, y_i)\), using a \(3 \times 4\) matrix:

\[
\begin{bmatrix}
f x_s \\
f y_s \\
0
\end{bmatrix}
= \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_s \\
y_s \\
z_s
\end{bmatrix}
\]

(Eq. 2.9)

Because the matrix above maps a 4D space to a 3D space, there must be some non-zero vector that maps to \((0, 0, 0)\). This vector defines the null space of the projection. This null vector is \((0, 0, 0, 1)\). This mapping can be described linearly in homogeneous coordinates and more specifically it can be rewritten as:

\[
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}
= \begin{bmatrix}
f x_s \\
f y_s \\
0
\end{bmatrix}
= \begin{bmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_s \\
y_s \\
z_s
\end{bmatrix}
\]

(Eq. 2.10)

Where the 2D image coordinates are provided via:

\[
x_i = \frac{u}{w}
\]

(Eq. 2.11)
\[ y_i = \frac{v}{w} \quad \text{(Eq. 2.12)} \]

To sum up, the 2D image point \( p \) is expressed in homogeneous coordinates as:

\[ p = (x_i, y_i, 1) = \left( \frac{u}{w}, \frac{v}{w}, \frac{w}{w} \right) \quad \text{(Eq. 2.13)} \]

**B. Transformation from Lengths to Pixels**

After getting the 2D image coordinates of a 3D point in camera coordinates, the next step is to transform from lengths to pixels. Typically, pixel numbers are used for indexing the points on the projection plane (image plane). For that, a transformation from the real physical positions to pixel positions is necessary. This involves scaling in both directions, because the height and width of a pixel is not a standard unit (like 1mm), and translation, because the pixel indices used in an image, usually start from the upper left corner and not from the principal point.

![Figure 2.5: The image \((x_0, y_0)\) and camera coordinate system \((x, y)\), adjusted from Yannick Morvan’s\(^8\) notes.](image)

Considering the image plane as a grid, we define a coordinate system on this grid in some length units at first, supposedly in millimeters or else mm. Moreover, the coordinate system has its origin at the center of the grid - else image center - \((x_0, y_0)\) and the axes \((x_{\text{pix}}, y_{\text{pix}})\) are parallel to the camera’s \(x\) and \(y\) axes respectively. In order to express the image plane in the units of pixels (instead of mm units), we perform scaling. We multiply the \(x\) and \(y\) coordinates by the number of pixels per mm in the \(x\) and \(y\) direction, respectively. These variables are our scale factors and are denoted as \(k_x, k_y\). Note that the two scale factors are not always exactly the same value, but are usually very close.

The projection and scaling transformation now becomes:

---

\(^8\) http://www.epixe.com/research/multi-view-coding-thesisse8.html
The previous transformation may give positions in terms of pixel units, but it does not correspond to the pixel indices used in an image. Typically, the index (0, 0) is in the upper left corner of the image. Thus, we need a translation in order to get the correct pixel coordinates of the principal point (else image center). Also note that the principal point does not necessarily correspond exactly to the center of the image sensor. The placement of the sensor relative to the lens (which defines the optical axis and hence principal point) might not be very accurate.

Let \((x_0, y_0)\) be the position of the principal point in pixel coordinates. It may indeed be at the center of the image, but it is also usual to be found at a slightly different position, as described in the previous paragraph. Then, to get the position of a projected point \((x_i, y_i)\) in pixel coordinates, we need to add a translation component which is the pixel position of the principal point. Taking into account the previous, together with the Eq. 2.10 and Eq. 2.13, the projected point in pixel coordinates becomes:

\[
x_{pix} = k_x x_i + x_0 = k_x f \frac{x_i + z_s x_0}{z_s} \quad \text{(Eq. 2.15)}
\]

\[
y_{pix} = k_y y_i + y_0 = k_y f \frac{y_i + z_s y_0}{z_s} \quad \text{(Eq. 2.16)}
\]

The previous equations can be rewritten as:

\[
\begin{bmatrix}
k_x f & 0 & x_0 & 0 \\
0 & k_y f & y_0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & x_0 & 0 \\
0 & 1 & y_0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
k_x f & 0 & 0 & 0 \\
0 & k_y f & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\] \quad \text{(Eq. 2.17)}

The scaling factors together with the focal length produce the variables \(a_x\) and \(a_y\), known as scaling variables or “focal lengths” in \(x\) and \(y\) direction respectively:

\[
a_x = k_x f \quad \text{(Eq. 2.18)}
\]

\[
a_y = k_y f \quad \text{(Eq. 2.19)}
\]

The parameters \(a_x\) and \(a_y\), and \(x_0\) and \(y_0\), are in pixels. Next, from Eq. 2.17, Eq. 2.18 and Eq. 2.19 we get:

\[
\begin{bmatrix}
u' \\ v' \\ w'
\end{bmatrix}
= \begin{bmatrix}
a_x & 0 & x_0 & 0 \\
0 & a_y & y_0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_i \\ y_i \\ z_s \\ 1
\end{bmatrix}
\] \quad \text{(Eq. 2.20)}
Concluding, the image coordinates in pixel units are provided by:

\[ x_{\text{pix}} = \frac{u'}{w'} \quad \text{(Eq. 2.21)} \]

\[ y_{\text{pix}} = \frac{v'}{w'} \quad \text{(Eq. 2.22)} \]

**C. Shape of pixels, Skewness and Intrinsic Parameter Matrix**

Some additional parameters that are occasionally useful to take into account are the shape of the pixels and the skewness, denoted as \( \eta \) and \( \tau \) parameters respectively. Figure 2.6 provides a visual representation of the previous parameters.

The Eq. 2.20 after inserting the additional two parameters of pixels’ shape and skewness becomes:

\[
\begin{bmatrix}
  u' \\
v' \\
w'
\end{bmatrix} =
\begin{bmatrix}
  a_x & \tau & x_0 & 0 \\
  0 & \eta a_y & y_0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_x \\
y_y \\
z_s
\end{bmatrix}
\quad \text{(Eq. 2.21)}
\]

If the skew parameter is different than zero (\( \tau \neq 0 \)) this means that the image coordinate axes are not orthogonal to each other. For some examples of lens distortion consult Figure 2.7. Furthermore, if the

---

10 http://campar.in.tum.de/twiki/pub/Chair/TeachingWs04IOIV/09StereoCameraTracking.pdf
pixels are square ($\eta = 1$), the resolution will be identical in both $x$ and $y$ directions of the camera image coordinates. However, the pixels are usually considered as rectangles with resolution $k_x$ and $k_y$ in pixels per another length units, in $x$ and $y$ direction respectively, as mentioned in Section 2.1.2.B.

Usually, the pixels are considered square, hence Eq.2.21 becomes:

$$
\begin{bmatrix}
    u' \\
    v' \\
    w'
\end{bmatrix}
= 
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    x_s \\
    y_s \\
    z_s \\
    1
\end{bmatrix}
= 
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    x_i \\
    y_i \\
    z_i \\
    1
\end{bmatrix}
$$

(Eq. 2.22)

The 3x4 matrix in Eq. 2.22 can be decomposed as:

$$
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
= 
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
$$

(Eq. 2.23)

The 3x3 upper triangular matrix that is used for the projection of a 3D scene point onto the camera image plane in Eq. 2.23 is now defined as $A$.

$$
A = 
\begin{bmatrix}
    a_x & \tau & a_x \\
    0 & a_y & a_y \\
    0 & 0 & 1
\end{bmatrix}
$$

(Eq. 2.24)

After replacing Eq. 2.24 in the Eq. 2.23, we get:

$$
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
= 
\begin{bmatrix}
    a_x & \tau & x_0 & 0 \\
    0 & a_y & y_0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
= 
A\begin{bmatrix}
    I_3 & 0_3
\end{bmatrix}
$$

(Eq. 2.25)

From Eq. 2.22 and Eq. 2.25, we have:

$$
\begin{bmatrix}
    u' \\
    v' \\
    w'
\end{bmatrix}
= 
A\begin{bmatrix}
    I_3 & 0_3
\end{bmatrix}
\begin{bmatrix}
    x_s \\
    y_s \\
    z_s \\
    1
\end{bmatrix}
$$

(Eq. 2.26)

It can be easily observed from the Eq. 2.24 that the matrix $A$ depends on the intrinsic camera parameters -like focal length ($f$) via the scaling variables $a_x$ and $a_y$, skewness ($\tau$), principal point ($x_0$ and $y_0$)-, therefore, this matrix defines the intrinsic parameters of the camera. This is usually called the intrinsic parameter matrix for the camera and it has five degrees of freedom.
**D. Transformation from Camera Coordinates to World Coordinates**

Thus far, we have expressed the position of points in the camera and image coordinate system, respectively. Nevertheless, it is often useful to express the position of a point in a world coordinate system which is independent of the camera, because usually we want to refer to the position of an object, regardless of the camera position. Therefore, we need a way of transforming between the world coordinate system and the camera coordinate system. This can be achieved via a 4x4 transformation matrix consisting of a translation vector and rotation matrix.

![Figure 2.8: Representation of a camera coordinate system with respect to a world coordinate system, taken from Ramani Duraiswami’s cmsc828d notes.](image)

Figure 2.8 illustrates two 3D coordinate systems, one for the camera and another for a predefined world. Let $X$, $Y$, and $Z$ be the orthogonal axes of the world coordinate system and $O$ point its origin. Similarly, $x$, $y$, and $z$ are the orthogonal axes of the camera coordinate system and $C$ point is its origin (also known as center of projection). Consider point $M$ as a point in the scene. This point is expressed as $(X_s, Y_s, Z_s)$ in the world coordinate system and as $(x_s, y_s, z_s)$ in the camera coordinate system.

![Figure 2.9: Focusing on the translation and rotation between the two coordinate systems, expressed in the camera coordinate system, adjusted from Ramani Duraiswami’s cmsc828d notes.](image)

In Figure 2.9, the two coordinate systems are illustrated as viewed from the $y$ axis. Let the translation of the origin of the camera coordinate system ($C$) to the origin of the world coordinate system ($O$) be given by vector $T = (T_x, T_y, T_z)$, or else written as $CO$. Moreover, vector $CM$ connects point $C$ with the scene point $M$ and acts as a projection line from the scene point to the image plane through the
center of projection $C$. The previous vector can be expressed as the addition of vectors $\overline{CO}$ and $\overline{OM}$ as it follows:

$$\overline{CM} = \overline{CO} + \overline{OM}$$

$$x_s \hat{i} + y_s \hat{j} + z_s \hat{k} = T_x \hat{i} + T_y \hat{j} + T_z \hat{k} + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k}$$

(Eq. 2.27)

From Eq. 2.27 each camera coordinate component is:

- \( x_s \hat{i} = T_x \hat{i} + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \Rightarrow \)
  \[ x_s = T_x + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \hat{i} \]
  (Eq. 2.28)

- \( y_s \hat{j} = T_y \hat{j} + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \Rightarrow \)
  \[ y_s = T_y + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \hat{j} \]
  (Eq. 2.29)

- \( z_s \hat{k} = T_z \hat{k} + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \Rightarrow \)
  \[ z_s = T_z + X_s \hat{i} + Y_s \hat{j} + Z_s \hat{k} \hat{k} \]
  (Eq. 2.30)

The previous three equations lead to the following representation in matrix form:

\[
\begin{bmatrix}
  x_s \\
  y_s \\
  z_s
\end{bmatrix} =
\begin{bmatrix}
  T_x \\
  T_y \\
  T_z
\end{bmatrix} +
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  T_x \\
  T_y \\
  T_z
\end{bmatrix} +
\begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X_s \\
  Y_s \\
  Z_s
\end{bmatrix} =
\begin{bmatrix}
  X_s \\
  Y_s \\
  Z_s
\end{bmatrix}
\]

(Eq. 2.31)

In homogeneous coordinates the previous equation is expressed as:

\[
\begin{bmatrix}
  x_s \\
  y_s \\
  z_s \\
  1
\end{bmatrix} =
\begin{bmatrix}
  R_{11} & R_{12} & R_{13} & T_x \\
  R_{21} & R_{22} & R_{23} & T_y \\
  R_{31} & R_{32} & R_{33} & T_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X_s \\
  Y_s \\
  Z_s \\
  1
\end{bmatrix} =
\begin{bmatrix}
  R \\
  T
\end{bmatrix}
\begin{bmatrix}
  X_s \\
  Y_s \\
  Z_s \\
  1
\end{bmatrix}
\]

(Eq. 2.32)

Let the rotation applied to make the principal axis coincide with the Z axis of the world coordinate system be given by a 3x3 rotation matrix $R$. Then the matrix formed by applying the rotation followed by the translation is given by the 3x4 matrix $[R \ T]$ and is called the extrinsic parameter matrix.
Now, consider beginning the transformation from the world coordinate system to the camera coordinate system. The translation vector becomes \( \overrightarrow{OC} \), therefore changes direction. Furthermore, vector \( \overrightarrow{CM} \) that connects point \( C \) with the scene point \( M \) can now be expressed as the subtraction of vectors \( \overrightarrow{OM} \) and \( \overrightarrow{OC} \) as it follows:

\[
\overrightarrow{CM} = \overrightarrow{OM} - \overrightarrow{OC} \\
(x_i \hat{i} + y_i \hat{j} + z_i \hat{k}) = (X_s - X_c) \hat{i} + (Y_s - Y_c) \hat{j} + (Z_s - Z_c) \hat{k} \quad \text{(Eq. 2.33)}
\]

From Eq. 2.33 each camera coordinate component is:

- \( x_i \hat{i} = (X_s - X_c) \hat{i} + (Y_s - Y_c) \hat{j} + (Z_s - Z_c) \hat{k} \) \quad \text{(Eq. 2.34)}

- \( y_s \hat{j} = (X_s - X_c) \hat{i} + (Y_s - Y_c) \hat{j} + (Z_s - Z_c) \hat{k} \) \quad \text{(Eq. 2.35)}

- \( z_s \hat{k} = (X_s - X_c) \hat{i} + (Y_s - Y_c) \hat{j} + (Z_s - Z_c) \hat{k} \) \quad \text{(Eq. 2.36)}

The previous three equations lead to the following representation in matrix form:

\[
\begin{bmatrix}
X_s - X_c \\
Y_s - Y_c \\
Z_s - Z_c
\end{bmatrix} =
\begin{bmatrix}
R_{11} & R_{12} & R_{13} \\
R_{21} & R_{22} & R_{23} \\
R_{31} & R_{32} & R_{33}
\end{bmatrix}
\begin{bmatrix}
X_i \\
Y_i \\
Z_i
\end{bmatrix}
\]

\text{(Eq. 2.37)}
The Eq. 2.37 can be further summarized by Eq. 2.38 below. Furthermore, Eq. 2.39 shows the correspondence of the translation vector $T$.

$$x_{\text{cam}} = R(X_{\text{world}} - C) \quad (\text{Eq. 2.38})$$

$$T = -RC \quad (\text{Eq. 2.39})$$

Where $C$ is the vector $\overrightarrow{OC}$ expressed in world coordinate system. Finally, the previous equations are expressed in homogeneous coordinates as:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & -X_c \\ R_{21} & R_{22} & R_{23} & -Y_c \\ R_{31} & R_{32} & R_{33} & -Z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} = \begin{bmatrix} R & -RC \\ 0_r^T & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (\text{Eq. 2.40})$$

**E. Linear Transformation from World Coordinates to Pixel Coordinates**

After having linked the 2D image coordinates, the 3D camera coordinates and the 3D world coordinates separately to one another, it’s time to combine them altogether and coordinate the transformation matrices into a single matrix.

Combining the Eq. 2.26 and Eq. 2.40, we get:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = A \begin{bmatrix} I_3 & 0_3 \end{bmatrix} \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} = A \begin{bmatrix} I_3 & 0_3 \end{bmatrix} \begin{bmatrix} R & -RC \\ 0_r^T & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} = M \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} \quad (\text{Eq. 2.41})$$

Which can be simply written as:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = M \begin{bmatrix} X_s \\ Y_s \\ Z_s \\ 1 \end{bmatrix} \Rightarrow p = M \cdot P \quad (\text{Eq. 2.42})$$

The 3x4 matrix $M$ is usually referred to as camera calibration matrix. The complete camera transformation can now be represented as:

$$M = A \begin{bmatrix} I_3 & 0_3 \end{bmatrix} \begin{bmatrix} R & -RC \\ 0_r^T & 1 \end{bmatrix} = A \begin{bmatrix} R & -RC \\ 0_r^T & 1 \end{bmatrix} = AR \begin{bmatrix} I_3 & -C \end{bmatrix} \quad (\text{Eq. 2.43})$$
The calibration matrix $M$ has 11 degrees of freedom. The degrees of freedom are distributed as it follows: 5 from triangular internal parameter matrix $A$, 3 from the rotation matrix $R$ and 3 from $C$.

Hence, $p$ the projection of $P$ is given by:

$$p = AR\begin{bmatrix} I_3 & -C \end{bmatrix}P = MP \quad \text{(Eq. 2.46)}$$

As shown in Eq. 2.46 and since $M$ is 3x4, $P$ is expressed in 4D homogeneous coordinates and $p$ derived by $MP$ is expressed in 3D homogeneous coordinates. The exact 2D location of the projection on the camera image plane is obtained by dividing the first two coordinates of $p$ by the third, which is the scaling factor.

### F. Mathematical Formulation

Let $P_i$ be a point in the world and $p_i$ be a point on the camera image plane. Similar to Eq. 2.46, the following equation provides us with the transformation from world coordinates to pixel coordinates up to a scaling factor $s$ (also equal to variable $w'$).

$$sp_i = AR\begin{bmatrix} I_3 & -C \end{bmatrix}P_i = MP_i \quad \text{(Eq. 2.47)}$$

As mentioned in the previous paragraphs, matrix $A$ keeps the intrinsic camera parameters and matrix $M$ is the calibration matrix, consisting of both intrinsic and extrinsic camera parameters. The calibration matrix $M$ can also be expressed as:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \quad \text{(Eq. 2.48)}$$

As implied in Eq. 2.47, the calibration matrix $M$ can be approximated by the multiplication of two matrices: one containing the intrinsics (named $M_{\text{int}}$) and the other containing the extrinsics (named $M_{\text{ext}}$). This is expressed by the following equation:

$$M = M_{\text{int}}M_{\text{ext}} \quad \text{(Eq. 2.49)}$$

Furthermore, the multiplication of the calibration matrix by a point in world coordinates is equivalent to the 2D pixel coordinates, see Eq. 2.50. Note that the multiplication of $M$ and $P$ is equal to the multiplied by a scaling factor 2D camera point, see Eq. 2.47 above and Eq. 2.51 below.

$$\begin{bmatrix} x_{\text{pix}} \\ y_{\text{pix}} \\ 1 \end{bmatrix} \sim M \begin{bmatrix} X_s \\ X_s \\ X_s \\ 1 \end{bmatrix} \quad \text{(Eq. 2.50)}$$
\[
\begin{bmatrix}
  w' x_{\text{pix}} \\
  w' y_{\text{pix}} \\
  w'
\end{bmatrix}
= M
\begin{bmatrix}
  X_s \\
  Y_s \\
  Z_r \\
  1
\end{bmatrix}
\text{(Eq. 2.51)}
\]

### 2.1.3 Existing Camera Calibration Software

Most of the approaches for camera calibration follow the pinhole camera model. From a quick internet search, we found a number of camera calibration software, from which we decided to test four and see if any of them would be of use for calibrating our hardware. The existing camera calibration software is provided in separate subsections below.

**Bouguet**

As the name of the section implies, the author of this work is Jean-Yves Bouguet. This work follows Zhang’s solution for camera calibration, see [Zhang 99]. Bouguet’s software for calibration is offered in Matlab and in OpenCV software. According to Bouguet’s website\(^\text{11}\), the last update took place in 2010.

We used this software to calibrate the RGB camera of Kinect. The use of this software resulted in several pros and limitations. The advantages are that the software provides accurate results and that it is lighting independent. Nevertheless, this software appeared to have some disadvantages for us as well, which discourage its use. For example, one negative point, was the fact that several screenshots need to be taken beforehand. Furthermore, during the calibration process, the user had to click on all four corners of the pattern in every screenshot, which might, from one hand improve the accuracy of the algorithm, but on the other hand, makes the process manual. See Figure 2.11 for the process of calibrating the RGB camera with Bouguet software.

**GML**

Another piece of software for camera calibration, which is similarly based on Zhang’s, solution is GML. This software is written in C++ and its source code and .exe is offered. The same team also

---

\(^{11}\) www.vision.caltech.edu/bouguetj/calib_doc
automated Bouguet’s calibration software in Matlab. Their last update was in 2011. See GML website\(^{12}\) for more information.

Similarly to the previous software, GML is also light independent, since the user clicks on critical points. Note that this software offers the possibility of using more than one pattern of different resolution, aiming to improve accuracy even more. Nevertheless, again, the user needs to take screenshots beforehand and the whole process involves a lot of clicking, especially in cases where more patterns are used. The software is available only in Windows (.exe), otherwise, the user needs to go through the source code, which is Visual Studio\(^ {13}\) friendly. See Figure 2.12 for a screenshot of the GML software user interface.

![Figure 2.12: Testing GML camera calibration.](image)

Apart from camera calibration software, which basically deals with camera to world transform in terms of extrinsics, software calibrating a projector with respect to a camera also exists. Since our system does contain this hardware pair (the RGB camera of Kinect and the projector) we decided to evaluate this type of software as well.

**VIBOT**

This software is also based in Zhang’s approach for camera calibration. The software is available in Matlab and is basically an extension of Bouguet’s Matlab software for camera calibration, extended to support projector to camera calibration, using ray-plane intersection. This software was last updated in 2009. More information is available on the project’s website\(^ {14}\).

The projector to camera calibration approach followed by this team, is quite interesting. They used ray-plane intersection in order to find the 3D-2D pairs of point correspondences. Note also that they worked on extending Bouguet’s existing software. As expected, the software is lighting independent, since the screenshots are taken beforehand. Also, the same negative points apply here, as in Bouguet’s software (first described software in Section 2.1.3). An important remark which is worth mentioning is that we were able to use only one screenshot during the process of projector calibration due to a crash of the software when we continued with the second screenshot. This would probably be something minor, nevertheless it required going deeper to the source code. The initiative behind the testing of existing projects was the discovery of software that can be immediately applied to our case.

\(^ {12}\) graphics.cs.msu.ru/en/science/research/calibration
\(^ {13}\) http://www.microsoft.com/visualstudio/eng/visual-studio-update
\(^ {14}\) code.google.com/p/procamcalib
by spending the minimum amount of time. We concluded that due to the use of one only screenshot, the result was far from accurate. See Figure 2.13 for the procedure of calibrating the projector to the camera with VIBOT.

![Figure 2.13: Using VIBOT software. a) in this image the camera view is shown b) this image shows the position of the projector, the RGB camera, the physical and the projected pattern in the environment, but also the window on the laptop’s screen showing the RGB camera view. The RGB camera that is depicted in the right picture is Sony PS3 Eye camera which was then replaced by the RGB camera of Kinect.](image)

**Audet**

This software calibrates the projector to a camera using Zhang’s method as well. Nevertheless, this work introduces the insertion of fiducial markers to camera-projector calibration. Half of the markers are physically printed, while the other half are displayed using the projector. This software is written in Java, and a .jar executable is offered. It depends on Java SE 6 or 7, OpenCV 2.3.1, and freenect library of OpenKinect hacking for Kinect. Its latest update was in 2012. For more information visit Audet’s website.

This software is definitely automatic and no screenshots are needed. It uses continuous capturing from the camera and saves screenshots whenever possible (detection of a number of pairs of markers for deriving a homography). Nevertheless, this software relies on fiducial marker detection and the lighting conditions can easily influence the result. Furthermore, it is claimed that the user simply waves the calibration board and the software calibrates the projector to the camera in about 30 seconds. Nevertheless, it is worth mentioning that during our tests we did not experience such a performance. In order for the software to detect the markers, we had to increase their size about 4 times the default one, meaning that, instead of using 100 fiducial markers, which is the default for accurate measurements, we used only ¼ of them (25), so that the camera could successfully detect a number of markers, in order for the software to get one of the screenshots, necessary for the process. After increasing the size of the markers and decreasing their total number, there was a major drop in accuracy. See Figure 2.14 for Audet’s projector to camera calibration.

---

15 [www.ok.ctrl.titech.ac.jp/~saudet/procamcalib](http://www.ok.ctrl.titech.ac.jp/~saudet/procamcalib)
2.2 Real World as Common Reference for System Calibration

In the previous section, we analyzed the theoretical background of calibrating a camera, and we described four pieces of software that we tried for camera and projector to camera calibration. Due to the fact that none of the software fits our needs, either from the accuracy point of view, or due to the difficulty in finding the units they use, or the orientation of the coordinate system, or the platform and dependencies, we decided not to use any of the software described in the previous section. Since the calibration of the hardware is extremely important for the correct and smooth operation of the SAR system, we decided to use the previous theoretical background of camera calibration and develop our own software, where everything will start from scratch and will be clear for us, since we will make our own assumptions for units, orientation, etc.

Some requirements and hints for calibrating the hardware components of our system are:

- The tracker arrives pre-calibrated and we need to set a trained 3D model as reference (the world).
- Kinect consists of an RGB camera and a Depth sensor.
- Projector is regarded as the inverse of a camera.

The first requirement implies that the coordinate system, which is set as reference (world) for the tracker, has to become the reference coordinate system (world) for the other hardware components as well, in case we want the results from each device to be coherent with results from other devices. Figure 2.15 shows a sketch of the common world concept. Therefore, we reached the decision to calibrate all the hardware components to the same world, which will be the world that is first set in tracker’s SDK. The following sections -Section 2.3, Section 2.4 and Section 2.5- elaborate on the calibration of each hardware component with respect to this common world.
In order to be able to find and decompose the calibration matrix, 3D-2D pairs of points are necessary. The whole idea of point correspondences is illustrated in Figure 2.16. Imagine that the lower sketch is our view of the real world and the upper is the RGB camera’s view of the world. It is a fact that each 3D point of the real world has a corresponding point on the 2D camera image. By sampling a number of 3D - 2D pairs, we will be able to find the calibration matrix, and eventually, calibrate the hardware component.

2.2.1 Sampling

The first step of calibration is the sampling of 3D to 2D corresponding pairs of points. These pairs of points will be later on used to compute the calibration matrix and the intrinsic and extrinsic camera parameters respectively.

Real world 3D points sampling

First of all, in order to be able to sample real world 3D points, the real world has to be defined. For this reason, a point representing the world origin and a 3D Cartesian coordinate system are needed. Then, the sampling of 3D points becomes a matter of measurement in the real world. To accurately measure the 3D world points, and for ease of use, we constructed a 2D pattern, see Figure 2.17a. This
pattern consists of two sizes of squares: the first is 5cm x 5cm, and the second is 10cm x 10cm. By using white and black color for these squares, it is easy to alter them accordingly and construct a certain area which will be easy to measure, without actually having to measure it with an instrument (e.g. ruler), but by just identifying the size information of the squares. For measuring the world’s altitude, we use as an object a rectangular parallelepiped, or else, a cuboid like the one shown in Figure 2.17b (also described in Section C.3.3). The only restriction is that the points sampled with the use of the 2D pattern and the cuboid must be non-planar (seek for an explanation in Section 2.2.2).

![Figure 2.17: Calibration pattern and calibration tool for 3D points sampling. The first is used for accurately measuring the x and y coordinates and the second is used for measuring the height (z axis). a) 2D calibration pattern b) cuboid calibration tool c) use for 3D point sampling](image)

**Image plane 2D points sampling**

Regarding the sampling of 2D points of the image plane, this can be done relatively easy when using a camera, simply by clicking on the corresponding points in the camera view. In case of a projector, remember that its projection is modeled as the inverse projection of a pinhole camera and is treated as a perspective projection, similar to the camera model. Therefore, the procedure, followed in the case of a camera, can also be used for the projector as well.

### 2.2.2 Mathematical Formulation

From Section 2.1.2 we derived that, a 3D point in world coordinates, multiplied by the calibration matrix, gives us the 2D point in the camera image plane in pixel units, multiplied by a scaling factor like:

\[
\begin{bmatrix}
    s H \\
    s V \\
    s
\end{bmatrix}
= M
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
\]

(Eq. 2.52)
Least Squares Approach
We now follow the least-squares approach in order to find the calibration matrix from the pairs of 3D-2D points that were sampled. By multiplying $M$ with the 3D world point in homogeneous coordinates, we get the following three equations:

$$s = m_{11}X + m_{12}Y + m_{13}Z + m_{14}$$ \hspace{1cm} (Eq. 2.53)

$$m_{11}X + m_{12}Y + m_{13}Z + m_{14} - u m_{31}X - u m_{32}Y - u m_{33}Z - u m_{34} = 0$$ \hspace{1cm} (Eq. 2.54)

$$m_{21}X + m_{22}Y + m_{23}Z + m_{24} - v m_{31}X - v m_{32}Y - v m_{33}Z - v m_{34} = 0$$ \hspace{1cm} (Eq. 2.55)

In Eq. 2.54 and Eq. 2.55 we replaced $s$ according to Eq. 2.53. From the last two equations, Eq. 2.54 and Eq. 2.55, for 1-pair of points we get:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & -X_1u_1 & -Y_1u_1 & -Z_1u_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1v_1 & -Y_1v_1 & -Z_1v_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & -X_Nu_N & -Y_Nu_N & -Z_Nu_N & -u_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -X_Nv_N & -Y_Nv_N & -Z_Nv_N & -v_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$ \hspace{1cm} (Eq. 2.56)

And if we extend the previous for N-pairs of points, we get:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & -X_1u_1 & -Y_1u_1 & -Z_1u_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1v_1 & -Y_1v_1 & -Z_1v_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & -X_Nu_N & -Y_Nu_N & -Z_Nu_N & -u_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -X_Nv_N & -Y_Nv_N & -Z_Nv_N & -v_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$ \hspace{1cm} (Eq. 2.57)

The equation 2.57 can be simply written as:

$$L \cdot \alpha = 0$$ \hspace{1cm} (Eq. 2.58)

which is a homogeneous linear equation system. The goal is to minimize $\|L \cdot \alpha\|$. The matrix $L$ is 2n x 12 but has rank at most 11 (12 minus the scale factor). If the 3D points are nearly planar, it may have rank 11 but the solution might be poor. If the 3D points are planar, the rank is only 8, which can still be used if we can otherwise determine some of the parameters (e.g. external). Nevertheless using non-planar points compensates for many of the previous problems, therefore we suggest using non-planar points for solving Eq. 2.58. Since we have two equations and 12 unknowns, we can solve this with at least 6 known 3D-2D pairs.

Singular Value Decomposition (SVD) to solve the Homogeneous Linear Equation System
The solution $\alpha$ is unique up to a scale. The error of Eq. 2.58 scales with $\|L \cdot \alpha\|$, so the minimum error is $\alpha = 0$. We now introduce a constraint which has to be included when solving for $\alpha$. 

50
The constraint is:
\[ m_{31}^2 + m_{32}^2 + m_{33}^2 = 1 \quad (\text{Eq. 2.59}) \]

The space of homogeneous solutions to the least-squares problem is given by the vector of \( V \) corresponding to the smallest eigenvalue when applying Singular Value Decomposition to \( L \):
\[ L = UWV^T \quad (\text{Eq. 2.60}) \]

Thus, the calibration matrix \( M \) reshaped from matrix \( V \) is:
\[
M = M_{\text{int}} M_{\text{ext}} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} = \begin{bmatrix} \alpha_u r_{11} + u_0 r_{31} & \alpha_u r_{12} + u_0 r_{32} & \alpha_u r_{13} + u_0 r_{33} & \alpha_u T_x + u_0 T_z \\ \alpha_v r_{21} + v_0 r_{31} & \alpha_v r_{22} + v_0 r_{32} & \alpha_v r_{23} + v_0 r_{33} & \alpha_v T_y + v_0 T_z \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} = \begin{bmatrix} m_{11}^T & m_{14} \\ m_{21}^T & m_{24} \\ m_{31}^T & m_{34} \end{bmatrix} \quad (\text{Eq. 2.61})
\]

From this equation, we can extract the intrinsic and extrinsic parameters \( u_0, v_0, \alpha_u, \alpha_v, T \) and \( R \).

We begin with the testing of the constraint that was posed in Eq. 2.59. Since the norm of matrix \( M \) is equal to 1, we can calculate the scale factor or else the constraint variable, suppose it is named lambda \( \lambda \).
\[
\lambda = \|m_3\| = \sqrt{m_{31}^2 + m_{32}^2 + m_{33}^2} \quad (\text{Eq. 2.62})
\]

Then, we scale the matrix \( M \) by dividing with the scale factor lambda:
\[
M = \frac{M}{\lambda} \quad (\text{Eq. 2.63})
\]

And now we make sure that the new lambda \( \lambda' \) is equal to 1 (\( \lambda' = \text{norm(new_m_3)} = 1 \)). This is according to the constraint posed in Eq. 2.59. Moreover, assuming that the world coordinate system lies in front of the camera coordinate system, a variable \( g \) is introduced to retain the sign (\( g = \text{sign(new_m_34)} \)). Now, we continue decomposing \( M \) to find the parameters. From the above comment on the sign and the Eq. 2.61, it becomes clear that:
\[
T_z = g \cdot m_{34} \quad (\text{Eq. 2.64})
\]
\[
r_5 = [r_{31}, r_{32}, r_{33}] = g \cdot m_{34}' \quad (\text{Eq. 2.65})
\]

Now, we compute the following dot products to get the variables that form the principal point, \( u_0 \) and \( v_0 \), and the variables related to the scaling and focal length, \( \alpha_u \) and \( \alpha_v \), respectively.
\[ u_0 = m_1^T \cdot m_3 \quad \text{(Eq. 2.66)} \]
\[ v_0 = m_2^T \cdot m_3 \quad \text{(Eq. 2.67)} \]
\[ a_u = \sqrt{m_1^T \cdot m_1 - u_0^2} \quad \text{(Eq. 2.68)} \]
\[ a_v = \sqrt{m_2^T \cdot m_2 - v_0^2} \quad \text{(Eq. 2.69)} \]

We can now calculate the first and second rows of the rotation matrix:

\[ r_1 = g \frac{(u_0 m_3^T - m_4^T)}{a_u} \quad \text{(Eq. 2.70)} \]
\[ r_2 = g \frac{(v_0 m_3^T - m_4^T)}{a_v} \quad \text{(Eq. 2.71)} \]

We can also calculate the first and second elements of the translation vector:

\[ T_x = g \frac{(u_0 m_{34} - m_{14})}{a_u} \quad \text{(Eq. 2.72)} \]
\[ T_y = g \frac{(v_0 m_{34} - m_{24})}{a_v} \quad \text{(Eq. 2.73)} \]

The rotation matrix \( R \) obtained with the previous estimation procedure is not guaranteed to be orthogonal. Therefore, we calculate the rotation matrix that is closest to the estimated matrix (in the Frobenius norm sense). We decompose \( R \) using Singular Value Decomposition \( (R = UV^T) \) and then calculate the dot product between \( U \) and \( V \) \( (R = UV^T) \). By applying Singular Value Decomposition we get, from the following equation, \( U \) and \( V \):

\[ R = UV^T \quad \text{(Eq. 2.74)} \]

Finally, after calculating the dot product between \( U \) and \( V \), we get the closest estimated rotation matrix \( R \):

\[ R = U \cdot V \quad \text{(Eq. 2.75)} \]

The above procedure was based on Chaudhury et al. [10], Siddhant Ahuja’s notes\(^1\), Bryan S. Morse’s CS750 notes\(^2\), and Bimber et al. [05] publications for camera calibration. See [Chaudhury 10], Section III A, equations (16)-(23) and [Bimber 05], pages 331-334, for more information.

\(^1\) http://siddhantahuja.wordpress.com/2010/02/20/570/
\(^2\) http://morse.cs.byu.edu/750/lectures/lect04/calibration.slides.pdf
2.3 Tracker to World Calibration

As mentioned in Section 2.2, the tracker unit basically comes pre-calibrated. The only requirement is the training of a model consisting of retro-reflective markers, to represent the world and the setting of this model as reference for the tracker, after choosing a three dimensional point of this model as origin. See Figure 2.18 and Figure 2.19 for a visual explanation on how the previous are done. With these two steps, the entire system’s world coordinate system is set. The world coordinate system set in this step, is also used for projector to world calibration and RGB-D camera to world calibration.

![Figure 2.18: View from the two infrared cameras: the calibration pattern is visible in these two separate views.](image)

![Figure 2.19: PS-Tech tracker SDK. a) Training of the calibration pattem model with the help of the tracker’s SDK, b) Setting of the calibration pattern as reference coordinate system for the tracker.](image)

2.4 Projector to World Calibration

2.4.1 Orthographic projection of camera space for image plane sampling

For sampling the 2D points lying on the projector image plane that correspond to the 3D world points, we came up with the idea of regarding the projector as a virtual camera in the 3D graphics world. Note that we consider that the virtual 3D world and the real 3D world are the same for the projector. For better understanding of what it takes for an object of the virtual world to transform from its space to screen coordinates, Figure 2.20 is here to help.
The transformation of an object from its space to the screen space contains a number of sequential transformations, such as transform to world space, then to camera space and to canonical view volume. We consider the projection transformation from the camera space to the canonical view volume as orthographic. Figure 2.21 illustrates the orthographic and the perspective projection respectively. In the orthographic projection, the center of projection is at infinity, whereas in perspective projection, it converges at a point. In Figure 2.21, this point is the origin of the Cartesian coordinate system.

The parameters that we set for orthographic projection transformation from camera to screen space are according to the projector’s resolution, which is 1024x768 pixels. The left (l) and right (r) points are parallel to the x axis, the bottom (b) and top (t) points parallel to the y axis and the near (n) and far (f) parallel to the z axis respectively. Thus, the parameters of the orthographic projection matrix are:
After setting the camera space projection transform as orthographic, we sample 2D image plane points by clicking. From clicking, we get the 2D screen coordinates and we transform them to 3D world coordinates. For more details on how it’s done programmatically, see Appendix B.

For the sampling of 3D points of the real world, we assigned the $x$ and $y$ coordinates of specific points on the calibration pattern to letters of the alphabet from A to Z. Furthermore, the cuboid was used to assign the $z$ coordinate. Looking at Figure 2.23, the green circle with the label ‘1’ is the first 3D-2D pair of points that was sampled and corresponds to $(0,0,0)$ or else, the origin in world 3D coordinates in cm, and $(505,580)$ in 2D screen coordinates in pixels.

Keep in mind that we could sample again and again the same 3D points in the world, but we would always get different 2D screen coordinates, since they depend on the placement of the calibration...
pattern in the environment each time. Another example of 3D-2D pair sampling is the following: the calibration pattern point 'I', which is assigned to (-15, -5) and has height of 3cm, forms the (-15, -5, 3) point in 3D, else written as 'I3'. The corresponding point in 2D (459, 422) is sampled by clicking on the 3D world point that was just sampled. In that case, feedback of the click is given, at the 3D point that was clicked, by a green circle and a label showing the number of the 3D-2D sampled pair. In the case of 'I3', the label is ‘9’, implying that it is the 9th pair which is being sampled.

At this moment, since the correspondences between the projector pixels (2D) and the calibration landmarks (3D) have been established, the standard camera calibration methods can be adopted for the digital projector calibration as well. Thus, the sampled 3D points represent P, and the sampled 2D point represents p, (refer to Section 2.2.2). The procedure described in Section 2.2.2 is followed to derive the calibration matrix and the intrinsic and extrinsic parameters.

2.4.2 Conversion from Computer Vision to Computer Graphics

After applying the mathematics described in Section 2.2.2, the result is a calibration matrix, intrinsic and extrinsic parameters. This output has to be translated to computer graphics. There is a correspondence between the calibration results from computer vision to the 3D viewing in computer graphics. In 3D graphics, we need to transform a 3D scene into a 2D image in order to display it. The projection matrix used to convert the eye coordinates to the clip coordinates is used for the projection transformation. The pipeline of the 3D viewing in OpenGL is shown in Figure 2.24.

**Figure 2.24:** Pipeline of 3D viewing in OpenGL taken from songho.ca.\(^\text{18}\)

**Internal Parameters and View Frustum**

According to [Knudsen 97], the internal camera parameters correspond to the view frustum in computer graphics. More specifically, the view frustum can either be orthographic, as shown in Figure 2.25 on the left, or perspective, as shown in the same figure on the right. Since we consider the projector as a virtual camera adhering to pinhole camera model, the perspective viewing frustum applies in our case. The frustum parameters are provided below in more detail.

**Figure 2.25:** Frustum in OpenGL: the left is the orthographic viewing frustum and the right is the perspective viewing frustum, both taken from songho.ca.

\(^{18}\) http://www.songho.ca/opengl/gl_transform.html
Frustum Parameters

- **left, right**: they specify the coordinates for the left and right vertical clipping planes
- **bottom, top**: they specify the coordinates for the bottom and top horizontal clipping planes
- **near, far**: they specify the distances to the near and far depth clipping planes

The viewing volume is used to clip the objects that lie outside of it. The four sides and its top and bottom correspond to the six clipping planes, as shown in Figure 2.26. The (left, bottom) - near and (right, top) - near specify the points on the near clipping plane that are mapped to the lower left and upper right corners of the window, assuming that the eye is located at (0, 0, 0). The far parameter specifies the location of the far clipping plane. Both near and far must be positive.

![Figure 2.26: Projector viewing frustum, taken from The Red Book](http://fly.cc.fer.hr/~unreal/theredbook/chapter03.html)

The View Frustum in this case describes a matrix that produces a perspective projection. The matrix has the following form:

\[
\begin{bmatrix}
2n & 0 & r+l & 0 \\
\frac{r-l}{r-l} & 0 & t+b & 0 \\
\frac{t-b}{t-b} & 0 & -(f+n) & -2fn \\
0 & 0 & f-n & f-n \\
0 & 0 & -1 & 0 \\
\end{bmatrix}
\]

(Eq. 2.76)

From projector to world calibration, among others, we get the intrinsic parameters. These parameters can be used in order to find the **near, far, left, right, top, bottom** parameters, which form the perspective projection matrix that converts the eye coordinates to the clip coordinates, as described previously. The intrinsic parameters that we get from the calibration are: the focal length \(f\), the principal point \((u_0, v_0)\) and the scaling coefficients \(\alpha, \beta\). These parameters, the projector’s resolution (1024x768) and the assumption that the far plane is at 10000, gives us enough input to compute the frustum parameters by using the procedure defined in Table 2.1.
These parameters will be then used to form the perspective projection matrix in computer graphics. See Appendix B for the respective code snippet.

**External Parameters and Model View**

From the calibration, we get a 3x3 rotation matrix and a 3x1 translation vector as output, which compose the extrinsic parameters of a calibrated camera (projector in this case). The rotation matrix, combined with the translation vector, form a 4x4 transformation matrix (after adding the \([0 \ 0 \ 0 \ 1]\) vector). This 4x4 matrix corresponds to the Model View in computer graphics. See Appendix B for the respective code snippet.

After setting the Model View and the View frustum according to the afore-mentioned, we can consider that the projector is set as a virtual camera to the 3D virtual world that corresponds to the 3D real world. From this moment and on, since the world is known, we can project everywhere in this world. The Figure 2.27 shows the projector’s view of the virtual/real world.

![Figure 2.27: Projection at specific world points. Image on the left (a) depicts projector’s view of the virtual world. The image on the right (b) shows how the projection looks in the real world. Note that the virtual and the real world coincide.](image)

Figure 2.28 provides a more detailed representation of the projection. In this case, a 5cm x 5cm grid is projected at level zero of the world (\(z = 0\) in world coordinates). From this image, we can see that the grid is well aligned to the calibration pattern, which tells us that the projector to world calibration was successful. Section 1.2.1 of Part III provides more details about the calibration performance.
Figure 2.29 displays the projection of a sphere accompanied with the label ‘13’ at a specific world point (Figure 2.29a) and the projection of a 3D rectangular grid of the dimensions of the calibration tool, else known as calibration cuboid (Figure 2.29b).

2.5 RGB-D Camera to World Calibration

The RGB-D camera to world calibration consists of three parts. In the first part, the conventional camera calibration takes place, according to the pinhole camera model, as described in the previous sections. The second part involves the alignment of the RGB camera with the Depth sensor. These two ‘cameras’ have a predefined distance and orientation from each other, set by factory.
Nevertheless, there is still a need for an alignment that corresponds to the 2D RGB camera coordinates with the 3D Depth camera coordinates. The OpenNI software, which is being used for Kinect development, provides the alignment between the RGB and the Depth cameras. Nevertheless, this alignment is handled internally and the conversion between the two coordinate systems is unknown; hence, there is a need to compute the transformation between the 3D Kinect coordinates and the 3D world coordinates. The third part contains the transformation from the 3D Kinect coordinates to the 3D world coordinates.

2.5.1 RGB camera to real world Calibration

The first step for calibrating an RGB camera to the world is the sampling of 3D-2D pairs of corresponding points. For sampling the 3D world points, the calibration pattern is used in the same manner as previously (see Section 2.2.1). On the other hand, the sampling of 2D points is much more straightforward, since we can access the 2D coordinates of the camera image plane, simply by clicking on the window showing the camera view. Software that enables the sampling of the 3D-2D pairs of points was developed. Figure 2.30 shows a screenshot during the operation of this software for 3D-2D points sampling. The user inserts the 3D point via keyboard and then clicks on the inserted point as shown through the window, via the mouse. A legend is added below the viewer to keep the user informed about the progress with informative messages, such as guidelines on what to do, the number of 3D world points that were inserted and the number of 2D points that were clicked.

![Software for sampling 3D-2D point correspondences](image)

Figure 2.30: Software for sampling 3D-2D point correspondences. The user inserts via keyboard the 3D real world point in the form of letter and number e.g. ‘I3’ and then the user clicks on the respective point on the window of the RGB viewer. The legend displays messages informing the user about the next action.

After gathering approximately ten pairs of 3D-2D corresponding points that cover a certain area, these pairs are used for computing the calibration matrix and the intrinsic and extrinsic RGB camera parameters, by applying the math of Section 2.2.2 once more.
2.5.2 Transform from 3D RGB-D camera to real world 3D Cartesian Coordinate system

Since in many cases we want to convert a 3D point from Kinect back to real world 3D coordinates, it is necessary to calculate the transformation between these two coordinate systems. For computing this transform, we make use of the previous calibration (first part) and the OpenNI functionality for 2D RGB to 3D Depth alignment (second part). We slightly modify the previous software to only query the real world 3D points from the user (e.g. C7 and enter via keyboard) and draw the corresponding 2D points automatically, based on the computed calibration matrix. Then the 2D is converted to Kinect device 3D, with the help of OpenNI. At this stage, we sample 3D world to 3D Kinect device point correspondences. In Figure 2.31, the image on the left (Figure 2.31a) shows the software before the start of 3D-3D sampling, while the image on the right (Figure 2.31b) shows the software after having sampled 26 pairs of 3D world to 3D RGB-D device coordinates. Note that these two images are taken at different times (the system setting is different).

![Figure 2.31: Software for sampling Kinect 3D and World 3D point correspondences: a) Beginning of sampling, b) End of sampling.](image)

After the sampling of a number of pairs of points that are different from the ones used for RGB to world calibration, we are ready to compute the 3D RGB-D to the 3D world transformation. For this, a similar solution to the one described in Section 2.2.2 is applied. The approach is given below.

**Coordinate System Transformation**

Let $Y$ denote the 3D world coordinate system and let $X$ denote the Kinect device 3D coordinate system. The transformation between $X$ and $Y$ can be achieved through the following equation:

$$Y = R \cdot X + T$$

(Eq. 2.77)

where $R$ is the rotation matrix and $T$ is the translation vector. If we suppose that we have $N$ pairs of 3D points (device’s and world’s respectively) then these points can be written as vectors:

$$X = [X_1, X_2, \ldots, X_N]^T \quad \text{and} \quad Y = [Y_1, Y_2, \ldots, Y_N]^T$$

where $X_i = (x_{i1}, x_{i2}, x_{i3})$, $X_N = (x_{N1}, x_{N2}, x_{N3})$ and $Y_i = (y_{i1}, y_{i2}, y_{i3})$, $\ldots$, $Y_N = (y_{N1}, y_{N2}, y_{N3})$
The equation Eq. 2.77 for the mean value of all the pairs of points \((X_{\text{mean}}, Y_{\text{mean}})\) becomes:

\[
Y_{\text{mean}} = R \cdot X_{\text{mean}} + T \quad \text{(Eq. 2.78)}
\]

Solving Eq. 2.78 for \(T\) and replacing in equation Eq. 2.77 we get:

\[
Y - Y_{\text{mean}} = (X - X_{\text{mean}}) \cdot R \quad \text{(Eq. 2.79)}
\]

Which can be rewritten as:

\[
\begin{bmatrix}
y_1 - y_{\text{mean}_1} \\
y_2 - y_{\text{mean}_2} \\
y_3 - y_{\text{mean}_3} \\
\vdots \\
y_{N} - y_{\text{mean}_N}
\end{bmatrix}
= \begin{bmatrix}
x_1 - x_{\text{mean}_1} \\
x_2 - x_{\text{mean}_2} \\
x_3 - x_{\text{mean}_3} \\
\vdots \\
x_{N} - x_{\text{mean}_N}
\end{bmatrix}
\begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
\vdots \\
r_{N-1}
\end{bmatrix}
\]

And expressed as:

\[
A \cdot Z = b \quad \text{(Eq. 2.80)}
\]

\[
A = \begin{bmatrix}
x_1 - x_{\text{mean}_1} & x_2 - x_{\text{mean}_2} & x_3 - x_{\text{mean}_3} \\
x_1 - x_{\text{mean}_1} & x_2 - x_{\text{mean}_2} & x_3 - x_{\text{mean}_3} \\
\vdots & \vdots & \vdots \\
x_1 - x_{\text{mean}_1} & x_2 - x_{\text{mean}_2} & x_3 - x_{\text{mean}_3}
\end{bmatrix},
\]

\[
b = \begin{bmatrix}
y_1 - y_{\text{mean}_1} \\
y_2 - y_{\text{mean}_2} \\
y_3 - y_{\text{mean}_3} \\
\vdots \\
y_{N} - y_{\text{mean}_N}
\end{bmatrix},
\]

and

\[
Z = \begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
\vdots \\
r_{N-1}
\end{bmatrix}
\]

which is an over-determined problem. Least squares approach is used to solve the linear equations system. After finding \(Z\), which is actually the rotation matrix, we replaced \(R\) in equation Eq. 2.78 to find the translation vector \(T\).
**Setting Local Coordinates in the point cloud**

The previous procedure, for finding the transformation from the RGB-D 3D device coordinates to the world 3D coordinates, requires sampling of a large number of pairs of points. This might end up being tedious. In Chapter 4, we need to get the transformation from RGB-D camera coordinates to the coordinate system of the cheese platter. If we used the previous approach, we would have to sample a number of 3D-3D point correspondences and then we would need to translate the real world to the origin of the cheese platter coordinate system. To do that, we need to manually measure the translation and we foresee that this would induce some error. Since in Chapter 4 we need a quick alternative that can be used to get the transformation within a second, we came up with an additional solution for this transform. The approach is provided in the current subsection.

In Chapter 4, an RGB-D camera (Kinect) is used to get a point cloud. The points of the sampled cloud are given in RGB-D coordinates. This is a problem for the scanning procedure, since the tracker provides information about the position and orientation of the object, with respect to a different coordinate system (cheese platter’s). In order to proceed with the scanning, we need to make the tracker and the RGB-D camera use the same coordinate system. For that, we came up with a quick solution that does not require camera calibration.

First of all, the cheese platter model is set as reference for the IR tracker. This means that the cheese platter is now the world for the IR tracker, and the information about the position and orientation of other tracked devices will be provided with respect to the cheese platter.

The idea is now to find the transformation between the Kinect coordinate system and the cheese platter coordinate system. This could also be envisioned as setting a local coordinate system in the point cloud, in our case, at the cheese platter. As the yellow diagram in Figure 2.32 shows, at first, two 3D points need to be specified on the cheese platter surface. From this, a calculation of the local coordinate system takes place. The process continues by finding the transform between the Kinect and the cheese platter. When the transform is found, it is saved to a file to be used later for scanning purposes (see Chapter 4).

![Figure 2.32](image)

**Figure 2.32:** Diagram of the steps followed for computing the transformation.

Figure 2.33 shows the procedure from a user perspective. The necessary user input is two 3D points, which are provided by clicking onto their respective position inside a window of Kinect view. Figure 2.35 and Figure 2.36 show how the end result looks like.
Figure 2.33: a) The user needs to click first on point A and then on point B. After this, the computation of cheese platter’s local coordinate system takes place automatically, b) Window with axes showing the Kinect’s view. The user clicks on the first specified point A represented by a black sphere. The same applies for clicking on point B.

The automatic calculation of the local coordinate system in the point cloud is given in 8 steps. These steps are demonstrated in Figure 2.34. In short, the user provides two 3D points lying on the cheese platter surface, point ‘A’ and point ‘B’ respectively.

Figure 2.34: Analysis of Local Coordinate System computation.
These two points form a line segment (Figure 2.34a). As a next step, the midpoint of this line segment is computed. This produces point ‘C’, which will be the origin of the local coordinate system (Figure 2.34b). Then, from points ‘C’ and ‘B’, the \( CB \) vector is computed, followed by the computation of its unit vector (Figure 2.34c and Figure 2.34d). The point ‘B’ is refined after the computation of the unit vector. This vector corresponds to the x-axis of the local coordinate system (Figure 2.34d). The surface of the cheese platter is planar, thus its plane equation is calculated with the help of Point Cloud Library (Figure 2.34e). After checking that the points ‘C’ and ‘B’ belong to this plane, the normal vector of the plane at point ‘C’ is calculated (Figure 2.34f). This vector corresponds to the z-axis of the coordinate system (Figure 2.34g). Finally, the computation of y-axis is done, by computing the cross product of x- and z- axes (Figure 2.34h).

After having found the local coordinate system of the cheese platter, we need to find the transform from this to Kinect’s coordinate system. Therefore, we set 4 pairs of 3D corresponding points, which are actually the points of the unit vectors of the axes. With the help of Point Cloud Library, we get the 4x4 transformation matrix from Kinect coordinate to the local coordinate system (cheese platter). By applying this transformation to the sample point cloud, we observe the transform of the point cloud to the cheese platter coordinate system. The following figures (Figure 2.35 and Figure 2.36) show the sampled and the transformed to local coordinate system point cloud.

Figure 2.35: The result of thePreLoop: the upper image shows the point cloud sampled by Kinect from the viewing of Kinect in Kinect coordinates. The calculated local coordinated system of the cheese platter is visible (the one with the thinner axis lines, the other is Kinect’s coordinate system), the lower image shows the cloud transformed to the local coordinates.

Figure 2.36: Same result as previous from another perspective for better understanding.
2.6 Summary

This chapter provided a complete presentation of the steps that were followed in order to calibrate each hardware component with respect to a common world. At first, the theoretical background on how to calibrate a camera that conforms to the pinhole model is described. After taking into account the system’s requirements, and more specifically the most basic one, which is the requirement to set a world reference frame for the IR tracker hardware component, we set this particular world as common reference, in order to calibrate the rest of the hardware components as well. Therefore, the chapter provided details on the setting of the IR tracker, the projector and the RGB-D camera to this common world respectively. We have now reached the point where the hardware components of our SAR system are connected and are able to communicate correctly. In the next chapters, the system gains much more functionality than this.
Chapter 3: Interaction Design

This chapter contains the design and the development of means to support interaction with the SAR system. First of all, the spatial realm for interaction is defined based on the system setting. Then, after exploring the alternatives for user input to the SAR system, the most prevailing is being developed. The chapter continues with the design of user interface with the system. Finally, the chapter ends with design choices for supported functionality.

3.1 About Interaction

After having set all the hardware components to acknowledge a defined common world in order to achieve effective communication, we proceed with the design of the necessary components in order to be able to interact with the system later on. Therefore, this chapter is dedicated to the interaction design for our SAR system. The interaction design starts with the definition of the appropriate spatial realm for interaction, based on the layout of the environment in which the SAR system will function. The spatial realm, in which our SAR system belongs, will steer us into choosing the appropriate components for achieving effective interaction between the user and the system. The chapter continues with an exploration on the most suitable type of user input and interface, depending on the requirements of our SAR system, in Section 3.3 and Section 3.4. Finally, the functionality that accompanies the user interface is defined accordingly in Section 3.5. Apart from the reasoning behind the decisions that were taken for each component (e.g. user input), details on the development are being provided as well, in each section respectively.

3.2 Defining the Spatial Realm for Interaction

As stated by Elepfandt et al. in Section 2.2.1 of Part I, Fred H. Previc introduced four Spatial Realms for Interaction. Beginning from the closest and continuing with more distant, Previc’s spatial realms are: the Peripersonal, the Action Extrapersonal, the Focal Extrapersonal, and the Ambient Extrapersonal, as shown in Figure 3.1a. Let us remind you here the following: the peripersonal region is within hand reach, whereas in focal extrapersonal region, visual search and recognition of objects takes places, the action extrapersonal is for navigation, spatial orientation, and the ambient extrapersonal supports postural control and locomotion. Similarly, Edward T. Hall introduced the Personal Reaction Bubbles back in 1966 [Hall 66], differentiating the type of space based on the radial distance from the person. The reaction bubbles are, from the closest to more distant ones: the Intimate Space (0m-0.45m), the Personal Space (0.45m-1.2m), the Social Space (1.2m-3.6m) and the Public Space (3.6m-7.6m), as shown in Figure 3.1b.
Based on Previc’s scheme, we are in the peripersonal region, since the interaction with our system has to be within hand reach. The other regions are excluded for the following reasons: the action extrapersonal region implies movement of the user, and the other two would be too far for interaction with our SAR system. Hall’s scheme suggests that our interaction with the system should be within the personal space region, from 0.45m to 1.2m.

3.3 User Input

To accomplish interaction between the user and the system, the system requires some form of user input. In general, there are several ways to provide user input (e.g. speech, gaze, hand, tools, etc). According to the previous section, based on the setting of the system, it was concluded that the user will be interacting with the system within the peripersonal region, meaning that the user will provide input to the system within his/her hand reach. The user input that is investigated in this section, is hand, finger and pen, all belonging to the category of within hand reach active input.

According to our setup, the hand could be used as user input only for the selection of a relatively large object. Nevertheless, the concept accompanying the design and development of our system is that it is applied as a tool supporting rapid prototyping in industrial design engineering. A task performed during rapid prototyping, such as the design of a feature (e.g. button), requires more detail, which can’t be supported by hand as the only user input. One could suggest that the user input is provided via user’s finger tip(s). Due to the nature of the field of application, an approach like the use of hand(s) and/or finger(s) does not match well the design procedure. Note that industrial designers are used to a number of tools for design (e.g. pencils of various hardness). Design with their finger tips would never produce the result they want to and this new way of designing is expected to be awkward and not a practical idea.

http://en.wikipedia.org/wiki/Personal_space
Tracked Pen

We decided to use a tracked pen for input to our SAR system, since this form of input seems to be the most appropriate, compared to others. The designer is expected to use the pen in similar manner as existing tools, like pencils. Furthermore, due to the fact that accuracy is very important for design, precision and reliability during tracking is very important for us, thus we decided to attach retro-reflective markers on the pen and track it with the IR optical tracker of the system. Inspiration was also drawn from Marner et al. [09].

Construction

In order to use a tracked pen as user input for our system, we need to construct one. This sub-section describes the construction of our pen.

![Construction of a 3D crosshair](image)

First, for reliable tracking, we need to place the retro-reflective markers in positions that ensure the tracking of at least four markers at all times. This means that we need to think about the geometry of the object that carries the markers, but also the placement of the markers. We decided to create a 3D crosshair, and attach it to the top part of the pen, in order to place the markers, since the cylindrical surface of the pen is expected to be covered by the user’s fingers. The construction of the 3D crosshair is demonstrated in Figure 3.2.

After the construction of the 3D crosshair, we attached retro-reflective markers at specific positions onto it. During the stage of model training using the tracker’s SDK, we observed that the tracker had difficulties in detecting the retro-reflective markers. After viewing the scene from the IR camera views, we concluded that the material that was used to construct the 3D crosshair, had high levels of reflectance, and in many cases the retro-reflective markers were not at all visible.

This triggers the need to explore the effect that different materials have on tracking. The following figure (Figure 3.3) shows a small test for finding the optimal material to be used in the 3D crosshair, which includes the attachment of retro-reflective markers to a number of materials and the viewing from the tracker’s IR cameras. Our test showed that painting the styrofoam material in black yields the optimal result. Therefore, we painted the surface of the constructed 3D crosshair and reattached the retro-reflective markers. The result can be viewed in Figure 3.4.
Figure 3.3: a) Test sample of our different materials, to determine the optimal one to be used as a 3D crosshair on the top of the pen (A: paper tape, B: black duct tape, C: white styrofoam, D: styrofoam painted black), b) Styrofoam painted black (B) appears to be the optimal material, since it is not reflective and the retro-reflective marker on it is always being tracked. The retro-reflective markers used in A and C materials are totally invisible. Material B is better than A and C but is still inappropriate.

Figure 3.4: The constructed pen made out of Styrofoam painted black and retro-reflective markers.

Finally, after attaching a number of retro-reflective stickers, we decided to add two retro-reflective spheres for two reasons. First of all, for improving the tracking coverage. Secondly, the attachment of the smaller sphere to the front part of the cap of the pen (bottom), targets the use of that sphere as the tip of the pen, getting closer to the natural sense and usage of a pen.

### 3.4 User Interface

After having defined the user input, it is necessary to define the interface between the user and the system. In standard HCI, such an interface would be a Graphical User Interface (GUI), but as explained in Section 2.2.1 of Part I, our system lies somewhere in between, meaning that characteristics of Human Computer Interaction (HCI) and Natural Human Interaction (NHI) can be combined to yield a result that would be appropriate to a SAR system.

Since the application field of our SAR system is Rapid Prototyping in Industrial Design Engineering, it is expected that the users will expect to get some support for carrying out their practice. The
designers need options such as brush size, color, etc. Considering the setting and the user input, we feel that the most appropriate way to provide them with these options is through a menu.

**Spatial Menu**
The nature of the SAR system has an impact on the design of a menu as a user interface. In our first attempts for menu design, we focused on the shape, size and its mobility in space. In order to keep complexity at low levels, the menu must provide options in a clear way and the number of options must be limited to the necessary. Furthermore, due to the layout of the setting, a menu that can be dynamically moved in the environment provides a lot of flexibility. As a practical solution we use a rectangular card, covering an area of 10cm x 15cm. Retro-reflective stickers are placed on the card, converting it to a tracked device. This allows the construction of a menu that covers relatively a small area (fits everywhere) and, at the same time, is being tracked in space (dynamic). In the first iteration of the menu design, we developed a main menu page containing two basic options: loading of textures and a palette for drawing. The procedure was the following: the user selects an option of the main menu and that menu page changes to a subpage related to the option. For example, if the user had chosen the option ‘draw’, then the subpage would be the palette providing color options and buttons for adding a new line, clearing existing lines and going back button, the latter for menu navigation. In Figure 3.5, the first iteration of the menu design can be viewed.

![Figure 3.5: The first iteration of the user interface design. a) The main menu page containing the options ‘alter’ and ‘draw’ for loading textures and choosing color respectively, b) The subpage related to the ‘alter’ option, c) The subpage related to the ‘draw’ option.](image)

![Figure 3.6: The revisited menu design. One page only, providing options such as color, brush size, adding of a new line, removing drawings from the scene, loading of models, and saving the scene to a file.](image)
The pilot test that was carried out during the last stages of this project, made us revisit the user interface design and simplify it even more by using only one page for providing the user with the available options. This menu page contains options for changing the color and the brush size of the pen for design purposes and also, contains options which provide more high level functionality, such as clearing the scene (reset), saving the scene, adding a line and loading a model. The last menu design is depicted in Figure 3.6.

3.5 Functionalities

It is a fact that a user interface cannot stand alone without its supported functionality. Thus, during user interface design, the setting of its respective functionality is necessary. At this early stage of the system, simple functionality needs to be enabled, such as selection and feedback. Moreover, since we are constructing a system to support a designer for carrying out design tasks, the annotation functionality is necessary. Finally, the adding, deleting, loading and saving of objects are considered higher level functionalities. The following subsections provide more information.

3.5.1 Selection

A menu must always enable the selection of an option. For this to happen, a listener to an event must be established. Therefore, a condition has to be met to trigger this event. We enable this simple event by applying a simple condition: if the tip of the pen falls within a specific 3D interval, which is less than half centimeter away from the menu, then, the event of clicking on the option is set, and the variables get updated accordingly. For example, if someone approaches the area above a color option in less than a certain distance, the selection of the specific color is triggered and the color of the pen is updated. This simple condition enables us to achieve selection, even though there are no physical buttons to press.

3.5.2 Feedback

Since the user interface of our system lacks tactile input, there is a need to increase the awareness of the user concerning his/her selections. For this, a form of feedback has to be established. Since the nature of our system engages the rendering of graphics on the real world, visual feedback seems a rational choice. Moreover, in some cases, audio feedback could also be used, in order to inform the user about the system’s state.

Audio

As mentioned above, audio can be used to increase user’s awareness about system’s state. By audio, we consider any kind of sound, not only oral speech. In this project, we limit ourselves to simple sounds for user feedback. Later on, in Chapter 4, we use this form of feedback as a way to inform the user that a process has finished and that the system is waiting for the user to take action again.
**Visual**

For enabling smoother user interface interaction, we decided to use visual cues to increase awareness. One example is the pen tip feedback on the color that has been selected. The user selects a specific color and the sphere projected on the tip of the pen changes its color accordingly. The same can be applied for the brush size. The size of the projected sphere can change according to the brush size selection. Nevertheless, we decided not to use the specific idea for size feedback, because it would hinder the drawing process, since its maximum size would be too large and the drawing would be occluded by the size of the sphere. Figure 3.7 shows a large-sized white sphere projected on the tip of the pen, informing the user that the pen is being tracked and that no color has been chosen so far (white equals to blank in this case).

![Image](image_url)

**Figure 3.7:** Visual feedback provided on the tip of the pen for color selection.

### 3.5.3 Annotation

The annotation functionality is an indispensible part of our SAR system. Based on this functionality, the system enables design. Note that the annotation on an object is achieved by connecting sequential vertices with the use of a line strip. Specific areas of the scene can be defined as active for annotation functionality in cases the geometry is known. For example, we can enable annotations on the top of the table (ground of the world that was set in Section 2.2), simply by applying a condition for distance. This condition can be the following: if the z-coordinate of the vector describing the position of the tip of the pen, is less than 0.5cm then the annotation functionality is on. This means that vertices that fulfill the previous condition are used to create the line for annotation. See Figure 3.8a for an example of enabling annotations on the ground-level of the system’s world.

![Image](image_url)

**Figure 3.8:** Some use cases of annotation functionality. a) Keeping notes for assistance or communication reasons, b) Design on the bounding box of a complex surface.

Rapid prototyping involves the design of a physical prototype which evolves through the process to form a final product design. Therefore, taking as granted a physical object as a physical prototype, the system needs to enable annotation on complex surfaces. The nature of our system enables two ways
for drawing on complex surface. The first one is drawing on the objects at a certain distance. For this, the boundaries of the volume covered by the object is needed and not the explicit geometry. Think of it like drawing on the bounding box of the object. This can work only if the geometry is a box though. See Figure 3.8b. In order to be able to annotate every surface without restrictions, we need to know the geometry of the object we want to draw on. If we have access to the vertices of the surface of a virtual model, then a condition for annotation can be set. For example, if the Euclidean distance between the position of the tip of the pen and the nearest vertex of the model’s vertices is less than 0.5cm, activate the annotation state for this object.

In order to find the nearest 3D point (vertex) of the virtual model to the tip of the pen, we use the k-d tree for storing the vertices of the virtual model. Then, the 3D point representing the position of the tip of the pen is provided to a request for the nearest neighbor. The query returns the nearest neighbor which is the nearest vertex. Due to the fact that this search introduces some lag to the application’s performance, we decided to check for the nearest neighbor between the pen and the virtual model from 2cm and less distance between the pen and the bounding box of the complex object. The following figure (Figure 3.9) displays the result of the annotation of complex surfaces.

![Figure 3.9: Drawing on the vertices forming the surface of a virtual model. In this image the cube with the fiducial marker on one of its face is linked to a virtual model with smooth surface and the cube with the retro-reflective stickers is linked to a virtual model with complex (rough) surface. The annotation functionality is the same for both cubical objects. Nevertheless, the geometry of the surface seems to have an effect on the visual result.](image)

### 3.6 Summary

In this chapter the design and development of the components that will be used to enable interaction between the user and our SAR system have been provided. Since our system belongs to the peripersonal region, the design was restricted to within hand reach interaction. A tracked pen was decided to be the most appropriate tool for user input, since the system needs to enable annotation. Furthermore, as a user interface, a spatial menu is chosen to provide the user with all the necessary options. Finally, functionalities such as selection, feedback and annotation have been linked to the user input and user interface.
Chapter 4: 3D Scanner and Application

This chapter contains the application of the designed SAR system. This entails a 3D scanner in order to get the virtual model of a physical object and tools for supporting interaction with the scanned object in a rapid prototyping scenario. The first section contains a small discussion on the system’s application. Then, the steps that were followed in order to construct the 3D scanner are provided. The final section refers to the user interaction with the resulting model and the world.

4.1 About 3D scanner and application to RP

The use of a 3D scanner in order to get the virtual model of an object is something common in industrial design. Industrial designers make use of high precision scanners such as laser scanners in order to accurately scan an object of their choice. The high quality of the entire point cloud gathered from such scanners is a result of a lengthy search to find the best match of corresponding landmarks between sequential scans for accurate registration. This implies that when more time is spent to get a scan, the result is expected to have better quality. Although laser 3D scanning technology provides good results, we decided to use Kinect in the place of a laser scanner to explore its operation in such a scenario.

As mentioned in Section 1.2, the first part of our application is the design and the development of a 3D scanner. For scanning, we fuse two 3D tracking technologies, namely an RGB-D camera and an infrared optical tracker. Then, after getting the virtual model of a physical object via scanning, we are in the position to interact with it, using the developed tools (see Chapter 3) to perform rapid prototyping scenarios.

4.2 Getting the Virtual Model via Scanning

The concept of fusing these two 3D tracking systems to produce a scanner, considers that the RGB-D camera is used for sampling the scene point cloud and the IR tracker for getting the accurate transform of the object with respect to a reference. The procedure starts by placing the object on the cheese platter. Then, the RGB-D device samples a point cloud. This point cloud is further processed to produce the point cloud of the object that is being scanned. Moreover, information about the position and orientation of the object from the IR tracker comes in handy, in order to register the segmented point cloud. One could observe that there is a barrier at this procedure since the IR tracker and the RGB-D camera refer to different coordinate systems. We know from Section 2.3 of Part II, that the IR tracker has the real world as reference. Furthermore, in Section 2.5.2 of Part II, we transformed the RGB-D 3D coordinates to the real world 3D coordinates. In the same section we use a similar procedure to transform Kinect 3D coordinates to world 3D coordinates, in order to be able to use the information of the tracker correctly, which is much quicker than the first one.

After training a model of the object - which is about to be scanned- with the tracker’s SDK so that it can be tracked (PS-Tech tracker SDK diagram in green, Figure 4.1), and after getting the transformation between Kinect and cheese platter (Section 2.5.2), we are ready to proceed with the
The actual scanning application (known as theLoop). The following diagram - shown in purple - in Figure 4.1, demonstrates the process that is used to get a point cloud representation of the object. At first, a point cloud is sampled with Kinect. This point cloud is filtered and then the object that lies on the cheese platter is segmented. The segmented object is then transformed to the local coordinate system of the cheese platter, so that it is consistent with the position and orientation that is provided from the IR tracker with respect to the local coordinate system. This information is used to register the current instance globally.

![Diagram of the 3D scanner cycle](image)

**Figure 4.1**: The 3D scanner cycle. The diagram in purple depicts the scanning and registration loop which is repeated until a condition is reached. After this an entire point cloud is produced. The mesh of this point cloud is then computed.

The 3D scanner application is further analyzed in the next paragraphs. Regarding the setting, as mentioned earlier, the user has to place the object on the cheese platter as shown in the first image of Figure 4.2. The end of each processing iteration is signaled by a ‘beep’ noise (audio feedback). Then the user has to rotate the cheese platter by an angle of his/her choice, and press any key to let the system proceed to the next iteration. Figure 4.2 describes this process visually.

![Scanning process images](image)

**Figure 4.2**: The scanning process: the user places the object on the cheese platter (1), waits for the audio feedback (2), rotates the cheese platter by an angle (3) and presses a key (4).
**Sampling**
The first step of the 3D scanner loop is the sampling. The RGB-D camera, or else Kinect, samples a point cloud from the scene of 640x480 resolution. An example of such a point cloud is shown in Figure 4.3.

![Sampled point cloud captured from Kinect.](image)

**Filtering**
The next step considers filtering. In order to choose points from the sampled point cloud that are useful to us, we apply a pass through filter to isolate the region between 0 and 1 meter of the z-axis in Kinect coordinates. This poses the restriction that the cheese platter has to be within the region of 1 meter from the RGB-D camera. Furthermore, outlier removal is used to remove noise. For each point, the mean distance from it to all its neighbors is computed. By assuming that the resulted distribution is Gaussian with a mean and a standard deviation, all points whose mean distances are outside an interval defined by the global distances mean and standard deviation can be considered as outliers and are trimmed from the dataset.

**Object Segmentation**
In order to achieve object segmentation, we decided to detect and remove large planar surfaces. Plane fitting is often applied to the task of detecting common indoor surfaces, such as walls, floors, and table tops. In our setting, table tops, some parts of the floor and walls close to the capturing RGB-D device are expected to be the surfaces that need to be detected and removed. Finally, the circular surface of the cheese platter has to be detected and removed as well. SAmple Consensus (SAC) methods like RANSAC and models like planes can be combined in order to detect specific models and their parameters in point clouds. The Point Cloud Library\(^\text{21}\) offers the SAC method with models such as lines, planes, cylinders, and spheres. For our purposes, the planar model is appropriate. After the planar surfaces are detected and removed, the result we get is a segmented object, like the one

---

\(^{21}\) [http://pointclouds.org/](http://pointclouds.org/)
shown in Figure 4.4. For more information on SAC method, see sample consensus\(^{22}\) offered by Point Cloud Library.

![Image](image1.png)

Figure 4.4: This image shows the object that is being scanned, segmented.

**Local Coordinates Transform**

In the Section 2.5.2 of Part II, the transform from Kinect to cheese platter coordinates is computed. At this stage of the scanning process, this transform is being applied. The segmented point cloud is being transformed to the local coordinates of the cheese platter. The Figure 4.5 displays the result of this transform.

![Image](image2.png)

Figure 4.5: Object transformed to local coordinates (cheese platter).

**PostFiltering**

Before registering the object globally, we decide to apply an additional noise removal, in case there are some leftovers from the segmentation step. We ignore noise outside the borders of the cheese platter 20 cm x 20 cm by using a pass through filter. Finally, we re-apply the outlier removal method.

**Global Registration**

At this stage the point cloud is ready to be globally registered. For registering the segmented point cloud, the information from the IR tracker is necessary. The IR tracker provides the current position and orientation of the tracked object with respect to the coordinate system of the cheese platter. The position is represented by a translation vector and the orientation by a rotation matrix. These two form...

\(^{22}\) http://docs.pointclouds.org/trunk/group__sample__consensus.html
the transformation matrix. We apply the inverse transform to register the object to the coordinate system of the cheese platter. The following figure (Figure 4.6) displays the registered point cloud from two different views for better understanding.

The process that was just described continues until an ending condition is met. We set the completion of one round (360°) as an ending condition for the loop. The resulting point cloud is iteratively completed, since different angles of the object need to be sampled. Figure 4.7 displays an entire point cloud after one round over the z-axis of the cheese platter coordinate system. In this figure, the consequence of the low resolution of Kinect is visible. Furthermore, a relatively small displacement is observed.

**Meshing**

After acquiring the entire point cloud of the object, the goal is to convert it to a polygonal mesh for future insertion in our graphics world of a “cleaned up” version of the virtual model. After testing a number of algorithms that the Point Cloud Library offers for meshing, such as Convex Hull, Organized Fast Mesh, Greedy Projection Triangulation, Grid Projection and Poisson, we concluded that Poisson yields the best result for the above point cloud. Of course, geometry and noise of the point cloud have a major impact. Furthermore, the parameters of each meshing approach need to be set to satisfy the needs of the point cloud. For the surface reconstruction of the point cloud of the
cubical object, the following parameters were used for surface normals estimation and Poisson reconstruction [Kazhdan 06]. We set the normal estimation radius search to 0.01 meters and the Poisson depth to 9. For more information about the surface normals estimation, see the corresponding section offered by Point Cloud Library\textsuperscript{23}. Finally, we smoothed the resulting mesh with Laplacian Smoothing offered by VTK library. The result is demonstrated in Figure 4.8.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{mesh.jpg}
\caption{The resulting polygonal mesh after applying Poisson Reconstruction.}
\end{figure}

### 4.3 Interaction with the reconstructed Model and the World

This section considers the application of the SAR system to support Rapid Prototyping. After the scanning of a physical prototype and the reconstruction of its surface, the insertion of this virtual model into the graphics world follows. Revisiting the diagram shown in Figure 1.2 of Part II, we now have a more complete view of the SAR system’s operation which is demonstrated in Figure 4.9.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{diagram.png}
\caption{Diagram of the SAR system operation for application to a rapid prototyping task.}
\end{figure}

\textsuperscript{23} http://www.pointclouds.org/documentation/tutorials/normal_estimation.php
The system supports a single user. The user must stay close to the SAR system setting in order to provide input to the system via a tracked pen. Options for performing a rapid prototyping task, such as design of features on the virtual model of the freshly scanned object, are provided through the dynamic menu which acts as the user interface to the system. The user can select color and brush size for the tip of the pen and can annotate on the scanned object and use the ground planar surface to as an assistive surface for keeping notes. Note that the virtual model of the scanned object consists of a complex surface even when the object that is being scanned is relatively smooth, like a cube. See Section 4.2 for more information on that. Furthermore, the menu provides the option to clear the lines that have been inserted by annotating, and reuse the space. The user can load textures of a 3D-print object existing in the scene accompanied with a number of textures. At this moment this is used for viewing, as used in past as well (Section 1.1 of Part II). An interesting extension would be to annotate on that virtual model as well. Finally, the user is able to save the entire scene to a file and access it later, see Figure 4.10 what a saved scene looks like.

Figure 4.10: This is a view of the saved scene which consists of the scanned cubical object and the menu. The result of annotation on the scanned physical object is displayed.

Figure 4.11: This is a view of the saved scene which consists of a scanned cubical object, a constructed cubical virtual model, the menu and annotations on the ground level of the world and the two cubes respectively.
In the real world the aforementioned would look like the following figures. Figure 4.12 shows the result from interacting with the SAR system in the real world. The graphics are overlaid on the corresponding physical objects. The difference between Figure 4.12 and the previous figure is that Figure 4.11 depicts the information rendered on a computer screen, instead of the real world via a projector.

![Figure 4.12: This is another view of the scene depicted in Figure 4.11. This is how the scene looks like in reality.](image1)

Figure 4.13 shows the cube, which was scanned before, and a 3D-print object, which is the front part of a tractor. The 3D-print and its perfect virtual model (with some embedded features such as tractor lights) is a useful example to see what a CAD model looks like when projected on its corresponding complex object. This is the best result that can be achieved in terms of a physical prototype accompanied with a mesh containing a texture like the one displayed in Figure 4.13.

![Figure 4.13: The virtual model of the scanned cube is overlaid on the cube in pink and the virtual model or the 3D-print is overlaid on the front part of the tractor in red.](image2)
4.4 Summary

This chapter contains the application of the developed SAR system, which consists of two parts respectively. The scanning application, which is used to get the virtual model out of a physical object, and the application to RP, which entails the use of the tools that were design to support interaction with the virtual models overlaid on physical objects and the world, during a RP scenario. For the scanning application the RGB-D camera was fused with the IR tracker. Section 4.2 provided step by step the procedure to get a complete virtual model of an object being scanned. Moreover, Section 4.3 described the system’s application to RP and its possibilities.
PART III

CONCLUSIONS
Chapter 1: Evaluation

This chapter contains details about the evaluation process that was followed for the developed SAR system. It starts with a quick discussion on evaluation in SAR and continues with the system’s performance analysis with respect to hardware calibration and application operation. Then, details on the user evaluation process are provided. Finally, the chapter ends with the presentation of the recorded results and their interpretation.

1.1 About Evaluation

It is important to underline that there are several different forms of evaluation. An evaluation of a system can be strictly technical in order to assess the performance of the system or the quality of the offered functionalities. A characteristic example is the evaluation by Menk et al. [11]. They focus on the quality of projected images by testing how experts would adjust and choose appropriate projection intensities so that the real color is replicated. Another form of evaluation is the empirical study in a specific research domain, in order to identify the steps of a process along with their perspective requirements. For a relative example, see [Verlinden 07]. Finally, the most common form of evaluation is the formal user evaluation, where researchers test the usability, user experience and the performance of a prototype system prior to becoming a final product.

Only half of the publications, in Chapter 2 of Part I, included a form of user evaluation of the proposed solution. Unfortunately, formal evaluations are particularly rare. The most common form of evaluation in these papers is limited to initial user reactions and feedback provided by short interviews or observations, like in Wilson et al. [10], Bandyopadhyay et al. [01], Raskar et al. [06], Cao et al. [07]. Post-study questionnaires are also used in some works, though the results are either not statistically processed, like in Tonn et al. [08], or do not yield any statistically significant results; thus the focus lies on interviews, like in Jones et al. [10]. Forlines et al. [05], Harrison et al. [11] and Porter et al. [10], carried out a formal evaluation to test the usability and the performance of the proposed solutions and processed statistically the results.

In general, it can be derived from the afore-mentioned that SAR research lacks in evaluation. Careful evaluation has to be incorporated in each proposed solution. In this chapter, we present the evaluation of our SAR system, which consists of two parts. The first part is a technical evaluation to assess the system’s performance and the second part contains a user evaluation of the developed SAR system to assess user experience.

1.2 Performance Analysis

This section contains the analysis of the system’s performance with respect to hardware calibration and application to RP. Section 1.2.1 starts with projector’s calibration accuracy, continues with RGB-D camera calibration accuracy, and ends with the presentation of the total error. Finally, Section 1.2.2 provides details about the scanning and application performance.
1.2.1 Hardware calibration performance

The performance of the calibration processes is being discussed in the current section. More specifically, the first sub-section provides information about the projector’s calibration accuracy, the next sub-section discusses RGB-D calibration accuracy and finally, the total error of the RGB-D and projector cycle is being visually evaluated.

**Projector calibration accuracy**

In order to form a more concise opinion about the calibration accuracy, apart from the intuitive visual one, we decided to project a 5cm x 5cm grid to match with the calibration pattern, as shown in Figure 2.28 of Part II. From this match, we are able to measure the deviation of the grid from the pattern, for a number of points. Table 1.1 below demonstrates the results.

<table>
<thead>
<tr>
<th>Point in X and Y</th>
<th>E (25,15)</th>
<th>M (10,15)</th>
<th>G (0,15)</th>
<th>H (-10,15)</th>
<th>F (5,10)</th>
<th>N (15,10)</th>
<th>O (20,10)</th>
<th>D (25,0)</th>
<th>P (20,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviation X in mm</td>
<td>-3.5</td>
<td>-3</td>
<td>-1</td>
<td>1</td>
<td>-0.5</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>deviation Y in mm</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.1: Measuring the projector calibration accuracy. Twenty seven points were used to measure the deviation from x- and y- axis respectively. All the measurements took place at zero level (z=0).

<table>
<thead>
<tr>
<th>B (10,0)</th>
<th>OR (0,0)</th>
<th>A (-10,0)</th>
<th>J (-10,-10)</th>
<th>K (0,-10)</th>
<th>L (10,-10)</th>
<th>C (20,-10)</th>
<th>I (-5,-15)</th>
<th>V (-10,-20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.5</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>-0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>-0.5</td>
<td>-0.5</td>
<td>-0.5</td>
<td>0</td>
<td>-0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q (5,-20)</th>
<th>R (15,-20)</th>
<th>W (25,-20)</th>
<th>S (15,-30)</th>
<th>T (5,-30)</th>
<th>U (-5,-30)</th>
<th>Y (-10,-35)</th>
<th>X (0,-35)</th>
<th>Z (25,-35)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>-2.5</td>
<td>-1.5</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
</tr>
</tbody>
</table>

Table 1.2: Considering z=0, root mean square \( X_{rms} = \sqrt{\frac{1}{N}(x_1^2 + x_2^2 + \cdots + x_N^2)} \) deviation from X and Y calculated from Table 1.1.
**RGB-D camera calibration accuracy**

For measuring the performance of the RGB camera to world calibration, we created a program, where the user inserts a point of the real world (e.g. ‘I3’) via keyboard and, as a result, a red circle is drawn on the viewer. Figure 1.1 displays the result of the RGB camera to real world calibration. In both figures, the real world points that were inserted belong to the ground level of the world (z=0). Since the letters of the calibration pattern are used to insert the x- and y- real world coordinates, we measure the performance based on the deviation of the red circles from the corresponding corners of the calibration pattern’s squares.

![Figure 1.1: RGB camera to real world calibration performance.](image)

For evaluating the performance of the rest of the RGB-D calibration, which involves the 2D image coordinates to the 3D real world coordinates, we built a program, where we click on 2D image coordinates of the RGB camera viewer. Then, the point is being transformed in 3D device coordinates and the 3D device coordinates are transformed back to the real 3D world. By this procedure, we provide a 2D point, visually approximating the 3D point of the world in the viewer, and we get the corresponding 3D point in real world coordinates, after having tested the 2D-3D alignment of OpenNI and the 3D device to 3D world transformation. The results are shown in Table 1.3.

<table>
<thead>
<tr>
<th>Origin or 0</th>
<th>Measured</th>
<th>Real</th>
<th>Abs Diff</th>
<th>Measured</th>
<th>Real</th>
<th>Abs Diff</th>
<th>Measured</th>
<th>Real</th>
<th>Abs Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td>Z</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1.0955151</td>
<td>0</td>
<td>-2.1875901</td>
<td>0</td>
<td>2.18759</td>
<td>-0.27570474</td>
<td>0</td>
<td>0.275705</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10.152017</td>
<td>10</td>
<td>0.152017</td>
<td>0</td>
<td>2.44968</td>
<td>-0.6267386</td>
<td>0</td>
<td>0.62674</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>19.417301</td>
<td>20</td>
<td>0.582699</td>
<td>0</td>
<td>2.814693</td>
<td>-0.5331768</td>
<td>0</td>
<td>0.533177</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>24.881794</td>
<td>25</td>
<td>0.118206</td>
<td>0</td>
<td>3.304162</td>
<td>-0.5331768</td>
<td>0</td>
<td>0.533177</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>19.720959</td>
<td>30</td>
<td>0.279041</td>
<td>0</td>
<td>3.814693</td>
<td>-0.26276386</td>
<td>0</td>
<td>0.262764</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>9.909563</td>
<td>10</td>
<td>0.090437</td>
<td>0</td>
<td>1.764117</td>
<td>-0.11960101</td>
<td>0</td>
<td>0.119601</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>1.054365</td>
<td>0</td>
<td>1.054365</td>
<td>0</td>
<td>1.11526</td>
<td>0.107001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>-3.583963</td>
<td>-5</td>
<td>1.4160137</td>
<td>-16.461803</td>
<td>-15</td>
<td>1.461003</td>
<td>-0.1970544</td>
<td>0</td>
<td>0.197057</td>
</tr>
<tr>
<td>70</td>
<td>5.589922</td>
<td>5</td>
<td>0.589922</td>
<td>-21.66696</td>
<td>-20</td>
<td>1.66696</td>
<td>-0.3284356</td>
<td>0</td>
<td>0.328424</td>
</tr>
<tr>
<td>80</td>
<td>14.796563</td>
<td>15</td>
<td>0.203437</td>
<td>-21.61689</td>
<td>-20</td>
<td>1.61689</td>
<td>-0.4751506</td>
<td>0</td>
<td>0.475151</td>
</tr>
<tr>
<td>90</td>
<td>24.481768</td>
<td>25</td>
<td>0.518232</td>
<td>-22.54858</td>
<td>-20</td>
<td>2.54858</td>
<td>-0.7281142</td>
<td>0</td>
<td>0.728114</td>
</tr>
<tr>
<td>100</td>
<td>14.964502</td>
<td>15</td>
<td>0.035498</td>
<td>-31.041094</td>
<td>-30</td>
<td>1.041094</td>
<td>-0.22367086</td>
<td>0</td>
<td>0.223671</td>
</tr>
<tr>
<td>110</td>
<td>5.493243</td>
<td>5</td>
<td>0.493243</td>
<td>-31.104776</td>
<td>-30</td>
<td>1.104776</td>
<td>-0.307465</td>
<td>0</td>
<td>0.307465</td>
</tr>
<tr>
<td>120</td>
<td>-3.514477</td>
<td>-5</td>
<td>1.4855225</td>
<td>-31.445204</td>
<td>-30</td>
<td>1.445204</td>
<td>-0.22657527</td>
<td>0</td>
<td>0.226575</td>
</tr>
<tr>
<td>130</td>
<td>-8.510054</td>
<td>-10</td>
<td>1.481946</td>
<td>-35.87953</td>
<td>-35</td>
<td>0.87953</td>
<td>-0.30889687</td>
<td>0</td>
<td>0.308897</td>
</tr>
<tr>
<td>140</td>
<td>0.716061</td>
<td>0</td>
<td>0.716061</td>
<td>-35.986427</td>
<td>-35</td>
<td>0.986427</td>
<td>-0.26140794</td>
<td>0</td>
<td>0.261408</td>
</tr>
<tr>
<td>150</td>
<td>24.009415</td>
<td>25</td>
<td>0.990585</td>
<td>-36.297077</td>
<td>-35</td>
<td>1.297077</td>
<td>-0.33827785</td>
<td>0</td>
<td>0.338278</td>
</tr>
<tr>
<td>RMS</td>
<td>0.8253</td>
<td></td>
<td></td>
<td>1.9289</td>
<td></td>
<td></td>
<td>0.4311</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.3: Measuring the RGB-D calibration accuracy in cm. More specifically, this table involves transformations from Kinect 2D to Kinect 3D and Kinect 3D to World 3D. The World 3D to Kinect 2D is not included in the measurements.
**Total Error**

After the completion of the calibration cycle, we are able to observe the total error inserted by each process separately. In order to have a visual representation, we decided to create a 2D array of points overlaid on the image plane of Kinect RGB camera. Each point is further converted to 3D Kinect device point, then transformed to 3D world point and finally, provided as input to the projector, in order to project on the specific 3D real world point. Keep in mind that the Kinect RGB camera calibration error is not taken into account in this procedure, since we start from the Kinect RGB camera 2D points and not from the world. Nevertheless, we still get a taste of the error that is inserted in this cycle. Figure 1.2 shows the aforementioned. It is known that the image size of the RGB and the depth camera of Kinect differ. The image resolution of the depth camera is noticeably lower than for the RGB. This fact can also be observed in Figure 1.2, where points of the depth image are only projected (in blue spheres). Moreover, the figure shows a noticeable deviation in x-axis, demonstrated by the translation of the yellow spheres projected on the calibration tool.

![Figure 1.2: This image shows a view from Kinect’s RGB camera after having gone from 2D RGB camera image coordinates (array of 2D points shown in red circles) to 3D camera image coordinates (taken care by OpenNI) to 3D world coordinates. Furthermore, the resulting 3D world coordinates are projected back to the world in a blue-to-yellow colormap representing the height (z axis in real world).](image)

The previous procedure for displaying the precision of the calibration cycle resulted in the indirect realization of an active scene exploration and 3D reconstruction example. In other words, Kinect can be used for world scanning and the projector can be used to project the acquired 3D points and contribute to the reconstruction of the world. The following figure (Figure 1.3) displays and explains the aforementioned and shows an example of altitude visualization by assigning a gradient color from blue to yellow colormap to the z-axis of the world.
1.2.2 Scanner and application performance

The performance of the 3D scanner and the application for interacting with the mesh and the real world is being discussed in this section. We begin with the 3D scanner’s performance. During the scanning process, we identified a small displacement of the sequential point clouds during the global registration, in Section 4.2. We investigated it further by rotating the cheese platter model alone, and we observed that this is caused by the fact that the rotation of the z-axis of the cheese platter is not accurately set. Figure 1.4 shows the error as observed in tracker’s SDK. For this to be solved an accurately manufactured turntable can be used instead of the cheese platter.

Figure 1.5 displays the resulting point cloud from the scanning procedure. As we can see the points that are sampled from the side faces of the cube contain holes and the sampling is not dense, nor well aligned. Note that only a small part of Kinect’s resolution is used, since most of the points are left out during segmentation. This means that the resulting point cloud appears to have low quality. Even on the overlapping face of the cube (the one on top) the points are not smoothly aligned. This means that the resulting surface is expected to have complex geometry. In regions of the point cloud where holes
are reported, the drawing is expected to be hard and of low quality. This is a result of the absence of vertices (sampled points).

![Figure 1.5: Image displaying the resulting point cloud quality. The zoomed out version is shown on the low right corner. Holes on the sides and roughly aligned points are visible.]

We now continue with the performance of the application for interaction with the virtual model and the world. The quality of the point cloud has certainly an effect on the annotation functionality. The mesh resulting from the point cloud consists of a complex surface with a similar shape to the scanned object. Nevertheless, the virtual model deviates from the physical object, in complexity and geometry. This is expected to have an influence on the annotation experience because the user intuitively expects to draw smoothly on the physical object, where in reality he/she designs on the mesh of the scanned object which has complex geometry. The first image in Figure 1.6 displays the result of annotation on a smooth cubical surface (left model, precise cube) and on the rough surface of the scanned model (right model, rough cube). Although the drawing of lines is continuous, the view of the projector and the bumps on the surface of the virtual model (complex geometry) results in the appearance of discontinuities in the drawings. The second image in Figure 1.6 shows an example in the real setting.

![Figure 1.6: Drawing on objects. a) The result of drawing on a perfect cube is shown on the left cube and the result of drawing on the scanned model of the cube, from a specific view in the graphics world, b) The viewing of the result in the real world setting.]

91
Finally, for comparison reasons we also scanned a 3D-print of a human heart. The result from the scanning procedure can be viewed in Figure 1.7, where the virtual model is overlaid on the physical one. The deviation from the physical surface was from about 0.5cm to 1cm. Moreover, crucial details such as arteries were not scanned.

Figure 1.7: Adding the 3D print object of a heart in the Real world setting. The heart is regarded as complex object and the two cubes as simple objects.

1.3 User Evaluation

Apart from the evaluation on system’s performance, which was described in Section 1.2, a user study was also conducted in order to evaluate the system with respect to its usability, functionality and usefulness. The current section provides the objectives for conducting this type of evaluation, the materials that were used, the experiment setup, and finally, the results together with the corresponding analysis and interpretation.

1.3.1 Test Objectives

As mentioned in Section 1.1, we concluded that, in many cases, SAR research is lacking evaluation. Furthermore, we suggested that careful evaluation must be incorporated in each proposed solution. Therefore, at this point it is useful, and needed, to learn about the strong points and weaknesses of the developed SAR system. This resulted in the realization of a user study, the objectives of which are provided in this section.

Our application consists of two parts; the 3D scanning, and the interaction with virtual models and the world, as described in Section 4.2 and Section 4.3 of Part II. More specifically, both parts are evaluated, in terms of Usability, Functionality, Usefulness and User Experience. The participants perform tasks and then respond to questions, based on the experiences they gained. Therefore, the type of study is descriptive.
Regarding the 3D Scanner application, the test is performed with respect to the following features:

- sound as feedback about the scanning progress
- interaction with the scanner (how easy, how natural, how quick)
- learning curve
- technical difficulties
- satisfaction with the result

Apart from the 3D Scanner, an evaluation for the developed application on interaction with CAD model and the real world (material surface/physical model) is also crucial and needed. Since this part of the system’s application focuses on the annotation functionality - in order to support the design of features on physical objects-, testing several points related to it is a need. The first test is related to the effect of the virtual model’s (CAD and/or scanned model) quality on the drawing experience. The second test is related to the comparison of the difficulty level/difference in drawing on a simple or complex object respectively, thus examine whether the shape of the physical model has a significant effect on the drawing experience. Since the whole concept of the developed system is accompanied by using the ground level of the real world as an active assistive surface for annotation, testing its usefulness is also of great interest. Moreover, additional functionalities need to be tested and evaluated, such as saving the scene and reusing it later on.

Apart from the afore-mentioned, testing the user interface and the tools that were designed to enable interaction between the user and the system is also of great importance. More specifically, the following topics were evaluated:

- feedback on the tip of the pen
- the pen as means for user input
- the menu as interface to the system
- mobility of the menu
- the clicking on the menu/selection of options
- learning curve
- functionality

Finally, direct input from users, regarding the system’s usefulness is also important. Therefore, a main questionnaire was formed, with respect to the following questions:

- Does the system meet user expectations?
- Is the new SAR system useful? Would the users use it in real life?

1.3.2 Materials

The user study materials consist of three sets, corresponding to the each step of the experiment respectively; the pre-test questionnaire, the SAR system test and the post-test questionnaire. For more information about the pre-test and post-test questionnaires, as well as the experiment brief, see Appendix D.
The pre-test material is a brief questionnaire including general questions regarding age, gender, field of study and level of experience with SAR systems and Rapid Prototyping. The goal is to gather some general background information for each participant.

The SAR system test materials include the overall developed system; a projector, an infrared optical tracker, Microsoft’s Kinect, a computer running the developed software, the cheese platter, the menu, three physical objects, the constructed pen and the experiment brief (a double-sided A4 leaflet). See Figure 1.8 below.

![Figure 1.8: Some of the materials for the experiment.](image)

The post-test material is an extended questionnaire, measuring the system’s usability, functionality, usefulness, as well as user experience. The questionnaire consists of four sections. The first section (A) includes questions about the scanner application, the second (B) about interaction with the scanned model, user interface and tools, as well as questions related to the system’s functionality. In the third section (C), the participant is asked to provide feedback about the usefulness of the SAR system, while the fourth section (D) is dedicated to further remarks. In total, the questionnaire consists of 47 questions. For every question, the user is asked to make a selection from a 6-point Likert scale and a “Not Applicable” (N/A) option. The scale has the following options: Strongly Disagree, Disagree, Slightly Disagree, Slightly Agree, Agree and Strongly Agree. Each option corresponds to an integer number between 1 and 6 respectively. An even number of options was applied to the scale, in order to encourage users to express an opinion. Some of the questions are asked in reverse order, to eliminate faulty results from “blind-folded” questionnaire fill in.

1.3.3 Experiment Setup

The number of participants was 21 in total, 7 females and 14 males. The participants were students in Industrial Design Engineering of varying experience/knowledge (Bachelor, Master). Only three of participants were familiar with the developed SAR system. Most of the students were slightly familiar with the SAR field, as well as with the use of 3D scanners. Obviously, most of the students had medium to good familiarity with Rapid Prototyping procedures. The Figure 1.9 below shows the background information gathered by the pre-test questionnaire.
Figure 1.9: Background information on familiarity of participants with respect to a) SAR field, b) 3D Scanners, c) Rapid Prototyping, d) our System.

The experiment was held at the INSYGHTLab of the Faculty of EEMCS at Delft University of Technology. Sessions of 30 minutes per person were scheduled. The test was completed in three days. Each session started with the participant filling in the pre-test questionnaire (taking less than a minute). Afterwards, instructions over the SAR system test were provided, together with the A4 leaflet (experiment brief), explaining the five tasks. The tasks were divided in two categories, each linked to the part of the system application to be tested (3D scanner and interaction with the resulting model and the world). Each part lasted about 10 minutes, 20 minutes for the SAR system test in total. Once the tasks were complete, each participant filled in the post-test questionnaire.

The tasks
The first part of the test consisted of the scanning of an object (physical model), in order to test the 3D Scanner Application. During this task, the participant rotates the cheese platter in sequential steps to get a full scan of an object. The used object for this experiment was a cardboard-based cube, 8.2 cm x 8.2 cm x 8.2 cm. After each rotation, the system processes the input, and upon process completion, it notifies the participant to proceed with the next rotation, by providing audio feedback. This process is completed after a full 360° rotation, where it prompts the user to terminate this session. The output is a total point cloud of the scanned object (cube) and a polygonal mesh of the point cloud. The rest of the tasks are related to the interaction with the model resulting from the scanning procedure and the world, using the spatial menu and the tracked pen. The participants were asked to perform four tasks
in this part. For the first task, one more cube, identical to the scanned one, was introduced, which was fixed to a specific location within the world (static cube). The first task included the drawing of a star on the surface of both scanned and static cubes. The mesh, resulting from the scanning procedure, is projected on the first cube (scanned cube), while a CAD modeled mesh is projected on the second one (static cube). The second task includes keeping notes on the ground level of the world. For the third task, a complex object (in this case, a 3D-print of a human heart analog) -which was already scanned- is inserted in the world, while the static cube is removed from the scene. The scanned cube is now regarded as simple object and the heart analog as a complex one. In this task, the user is asked to draw simple and complex shapes (e.g. a circle as simple and a flower as complex) on both objects. For the final task, the user is asked to save the scene to a file.

1.3.4 Results, Data Analysis and Interpretation

For the exploration and processing of data, gathered from the user evaluation, we use SPSS\(^{24}\). After inserting all the data into SPSS and declaring the missing values from the ‘N/A’ option, we reverse back the statements that were posed in reverse order. Note that occasionally the statements in the questionnaire appear reversed in order to distinguish, as well as eliminate faulty results from “blindfolded” questionnaire fill in. For example, a question like “I found the scanning process difficult to carry out”, after reversing becomes “I found the scanning process easy to carry out”. Keep in mind that all data were recorded according to the following assertion: Strongly Disagree = 1, Disagree = 2, Slightly Disagree = 3, Slightly Agree = 4, Agree = 5 and Strongly Agree = 6.

3D Scanner

The following results are related to the first category of questions, which correspond to the scanning task. Most of the participants agreed that the audio feedback is a successful way to be informed about the progress of the scanning procedure (\(\mu = 5.29, \sigma = .784\)), and that the turning of the cheese platter felt natural during the procedure (\(\mu = 4.86, \sigma = .793\)). For the corresponding histograms, see Figure 1.10.

\[\text{Figure 1.10: Attitude towards audio feedback in (a) and turning of the cheese platter in (b).}\]

On questions related to difficulty, learning and speed, most participants agreed that the procedure was easy (\(\mu = 4.67, \sigma = 1.11\)) nevertheless there were some participants that slightly disagreed on this.

\(^{24}\) http://www-01.ibm.com/software/analytics/spss/products/statistics/
The results were also similar about speed. Most of them agreed that the scanning process was quick \((\mu = 4.76, \sigma = 1.136)\). Finally, the level of agreement slightly drops on the question related to learning \((\mu = 4.14, \sigma = 1.153)\). The afore-mentioned results are displayed on the following histograms in Figure 1.11.

The question of special interest is the one related to the satisfaction of the participants with the scanner’s result (the polygonal mesh constructed from the point cloud). Most of the participants did not agree on the statement “I am satisfied with scanner’s result” \((\mu = 2.9, \sigma = 1.221)\). Nevertheless, approximately one third of the participants’ total number was positive and agreed with the previous statement. The following figure (Figure 1.12) displays the results.

The scanning results indicate that the simple audio feedback fits in the process as a way to inform the user about the progress of the scanning procedure. Furthermore, the turning of the cheese platter felt natural for most participants. Only one person slightly disagrees with the above, which leads to a rather secure conclusion that these two design features of the scanning procedure achieved their goal. Concerning the difficulty of the scanning procedure, approximately \(\frac{1}{4}\) of the participants found it slightly difficult and the rest agreed that was easy. Concerning the learning curve, again, although the majority agrees that the scanning doesn’t require learning, there is a considerable number of
participants that disagree. 7 persons in total believe that the scanning procedure needs some learning. This is something reasonable, since the procedure works by repeating 3 sequential steps. For most of the users the procedure was quick, nevertheless 3 persons found it a bit long. In order to see if the experience with a 3D scanner had an effect on the opinion about the duration of the scanning, we used Chi-square analysis and we found that there is no significant difference between familiarity with 3D scanners and the response expressing opinion on the duration of the scanning procedure, \( \chi^2(12, N = 21) = 11.667, \ p = 0.473 > .05 \). Finally, most of the participants disagreed on the statement that they are satisfied with the scanner’s result. The results show that most of them disagree or slightly disagree, nevertheless there are two instances of strong disagreement, and 4 instances of agreement. Therefore, the Chi-square was repeated in order to detect if there is an effect of the familiarity with 3D scanners on the opinion on the scanner’s result. The result was negative. The same applies for the familiarity with Rapid Prototyping and familiarity with SAR.

**Interaction with the model and the world**

In this part of the user study, the participants had to complete four tasks. The questionnaire is divided into four sections, in order to measure attributes related to each task respectively. Moreover, since this part is related to the user interface and tools, as well as the system’s functionality, these two are evaluated in this section as well.

**Task 1**

The goal of the first task was to measure the effect of the scanner’s result (mesh) quality on the drawing experience. Furthermore, questions, related to the object’s mobility, precision of drawing and the accuracy of the scanner’s result are posed. Figure 1.13 shows these results in detail. The results show that the participants’ opinions do not converge. This is also apparent from the value of standard deviation, which is almost equal to 1.5. This behavior is noticeable in the following statements: “It was easier to draw on the static cube”, “the precision of drawing was pretty equal in both cases” and “I preferred to draw on the dynamic cube due to its mobility”. On the fourth statement, about accuracy not being an issue during rapid prototyping, the opinions converge a bit more, thus the standard deviation decreases to 1. Most of the participants disagreed with the statement. Finally, once again, most of the participants agreed that the mobility of an object makes them gain more perspective on future designs ( \( \mu = 5, \ \sigma = .447 \)).
Concerning the second part of the application - interaction with the model and the world - the data of Task 1 provide the following conclusions. As mentioned in the previous paragraph, the opinions diverge in this task. For the first statement which aims at a confirmation that the quality of the mesh plays a role on the drawing experience, was not the case. Even though the static cube was linked to a high quality mesh, approximately half of the participants found it difficult to draw on. 9 participants had the opposite opinion and agreed that it was easy. Nevertheless, we could interpret this result based on observation. The static cube was located near the border of the tracking frustum, and some participants experienced difficulties with the tracking of the pen. Furthermore, the static nature of the cube restricted the ability of users to manipulate the object while drawing. These two are expected to have influenced the result. Regarding the equalization of the precision of drawing, the participants disagreed on this statement, as expected. This shows that most participants realized that the precision differed. Nevertheless, there are 6 participants that found it pretty equal. Concerning the mobility of the cube, again in this question, most of the participants agreed that the mobility plays a large role in the drawing procedure. More than half of them said that they preferred to draw on the dynamic cube due to its mobility. Nevertheless, 7 participants disagreed, which reveals that the quality of the mesh is important. Concerning the last statement, most of the participants agree that accuracy matters in Rapid Prototyping. There are 5 people in total that have the opposite opinion, but still, they slightly disagree. This shows that in such tasks, only a small amount of inaccuracy is allowed.

**Task 2**

This task is about keeping a note on the surface of the table (ground level of world). The level of difficulty of drawing on this surface and the usefulness of this assistive additional surface in the setting are being measured. Most of the participants agreed that drawing on the table’s surface was easy ($\mu = 4.95$, $\sigma = 1.284$) and useful for the current setting ($\mu = 4.9$, $\sigma = .889$). The results are provided in Figure 1.14.
Task 2 tested the easiness and the usefulness of keeping notes on the top surface of a table, which is set to be the ground level of our system’s world. Results show that this was easy for most, but kind of difficult for three participants. Finally, except for two participants that slightly disagreed, the rest found this addition useful.

**Task 3**
The third task requires the use of two different physical models, one of low and one of high complexity. The low complexity (simple) model was the previously used 8.2 x 8.2 x 8.2 (in cm) cube, while the high complexity (complex) object was the 3D print of the human heart analog. Both models were scanned, and each mesh was projected on its corresponding physical model. The user’s task was to draw a simple and a complex shape on both models. The participants answered four questions related to this task. The results are given in Figure 1.15 and Figure 1.16 below.

![Histogram](image1.png)  
**a)**  
![Histogram](image2.png)  
**b)**  

Figure 1.14: Top of the table surface as an active surface for annotation; a) The level of difficulty is low, b) It is useful.

![Histogram](image3.png)  
**a)**  
![Histogram](image4.png)  
**b)**  

Figure 1.15: Results of drawing a simple shape on a simple and a complex object, in (a) and (b) respectively.
According to the diagram in Figure 1.15a, 3/4 of the participants agreed that drawing a simple shape on the surface of a simple object was easy (\( \mu = 4.19, \sigma = 1.209 \)). Nevertheless, on the question about drawing the same shape on a complex object the prevalent opinion changes. Most of the participants found this difficult (\( \mu = 3.05, \sigma = 1.322 \)). Note that the standard deviation of the result on this question is around 1.3, and by looking at the histogram, we observe that 7 persons found this easy. Finally, on questions regarding drawing a complex shape on a simple object, half of the participants thought it was difficult, while the other half slightly agreed that it was easy (\( \mu = 3.24, \sigma = 1.513 \)). Concerning the last question about drawing a complex shape on a complex object, most of the participants found this hard (\( \mu = 2.52, \sigma = 1.209 \)). The results are shown below, on Figure 1.16.

Figure 1.16: Results of drawing a complex shape on a simple and a complex object, in (a) and (b) respectively.

The results from testing the drawing of simple and complex shapes on simple and complex surfaces are analyzed in this paragraph. Most agreed that drawing simple shapes on the object of simple geometry (the cube in our case) was easy. Nevertheless, five persons in total had the opposite opinion: 2 persons slightly disagreed and 3 persons disagreed. On the question about drawing a simple shape on objects of complex geometry (in this case, the 3D-print of a human heart analog), the majority found this difficult. Nevertheless, 7 participants believed that it was easy. In order to detect if there is a difference between drawing something simple on a simple object and drawing the same on a complex object, we use the Paired-Sample t-test. The test shows that there is a significant difference between these two variables, (\( t(20) = 3.294, p = 0.004 < .05 \)). Regarding the next question, the opinions were split in two, about the difficulty on drawing a complex shape on the surface of an object with simple geometry. Finally, concerning the question on drawing a complex shape on a complex object, it can be derived that in general the participants found this hard, except for 4 persons. We performed the Paired-Sample t-test for this as well, but there was no significant difference on the results.

**Task 4**
In this task, the user had to save the scene and then review the result. Questions about the importance of the clear and save button were asked in this section. Most of the participants agreed that the clear button enabled them to redo their design (\( \mu = 4.9, \sigma = .831 \)). Furthermore, they agreed that the save
button enables them to continue working on the design later (\( \mu = 5.57, \sigma = .507 \)), thus agreed that the save functionality enables the possibility to share their design with their colleagues (\( \mu = 5.14, \sigma = .854 \)). The results are presented in the following figure (Figure 1.17).

In the fourth task, the user had to save the entire scene, only by pressing a button. Then, the participant reviewed the result on the screen. Based on this experience and the use of the clear button during the drawing tasks, the participants provided feedback about the clear and save button. They agreed that due to the clear button they were able to redo their design. Furthermore, all agreed that the save button enables them to continue their work later, which shows that this is probably important for the participants. Finally, most agreed that due to the save button they could send their design to their collaborators and exchange ideas, except for 2 that slightly disagreed.

**User Interface and Tools**

This section contains the results regarding user input, the user interface and the enabled functionality, demonstrated in Figure 1.18. The users agreed that the feedback on the tip of the pen increased their awareness (\( \mu = 4.71, \sigma = 1.309 \)). They also agreed that the pen is appropriate for user input (\( \mu = 5.14, \sigma = .578 \)). Furthermore, the participants agreed that the mobility of the menu was convenient (\( \mu = 5.14, \sigma = .793 \)). Additionally, clicking on the menu was satisfactory (\( \mu = 4.7, \sigma = 1.081 \)).
On a question related to the interaction with the menu, despite the lack of the physical buttons, most of the users agreed that the interaction wasn’t affected, except for three of them who disagreed ($\mu = 4.86$, $\sigma = 1.153$). Although most of the participants agreed that the menu is an appropriate interface for the system, four participants had different opinion ($\mu = 4.7$, $\sigma = 1.218$), (Figure 1.18e). On the question about the learning curve of the menu, the results were classified in two groups. While nine of the participants disagreed that the menu required learning, the rest of them agreed ($\mu = 3.29$, $\sigma = 1.454$), (Figure 1.18f). Finally, most of the results were in favor of the menu’s ease of use ($\mu = 5.1$, $\sigma = 0.768$).

This part of the user evaluation was about the user input and user interface design. The first statement is about the sphere projected on the tip of the pen that acts as visual feedback for the tracking of the pen and the color selection. As expected, most agreed that the feedback on the tip increases their situation awareness. Furthermore, most agreed that the pen is an appropriate means for user input. The convenience of the mobility of the menu was also recognized. Concerning the clicking on the menu buttons, even though most agree that it was satisfactory, there are two participants that slightly disagree and one that disagrees. This shows that there is space for improvement. Concerning the menu as an appropriate interface for the system, again most of the participants agree, nevertheless four of them believe that there is an alternative. Finally, the results show that participants’ opinion split in two concerning the learning curve of the menu. For some, the menu requires no learning at all, but for
the others it does. This is something expected since some people are familiar with multi-touch technology, and might be quicker learners than others.

**System functionality**

This section provides the results on system’s functionality. The participants agreed that the system quickly detected the buttons they were trying to select (μ = 5.05, σ = .74) and that the tracking felt real-time (μ = 4.9, σ = 1.179). The majority of participants responded negatively to the experience of lag, nevertheless, seven of them had the opposite opinion (μ = 4.05, σ = 1.359). Finally, most of the users agreed that the graphics were displayed at the correct position (μ = 4.43, σ = .811).

Concerning the system’s functionality, the results show that the participants identified that user interface quickly detected the buttons they wanted to press and that the tracking felt real time. Regarding the lag, most of the users reported that there was no lag, nevertheless only 6 detected that there was a lag. During drawing on the surface of an object, the program scans the virtual model and searches for the nearest neighbor of the model to the tip of the pen. This introduces some lag, which was expected to be noticed. Finally, most of the participants agreed on the statement that the graphics were displayed at the correct position, nevertheless 6 of them disagreed. This is something not expected, and we think that these participants consider the displacement of the lines during drawing.
Nevertheless, this question aimed to measure the result of projector to world calibration, but apparently it was a bit vague.

**Usefulness of the SAR system**

This section contains an overview of the overall user opinion on the developed system and its value.

**Figure 1.20:** Results on usefulness of the system.
The Figure 1.20a shows that most of the participants would use the 3D scanner to get a mesh of a freshly made physical prototype. On the next question –whether or not they would make use of a better scanner for more accurate result–, the majority agrees, while, when participants were asked whether or not they would use this system for rapid prototyping, the responses were classified in two equal groups (Figure 1.20c). In addition, most participants agree that the system has the potential to support communication of ideas. Finally, on the last questions about usefulness, most of the participants consider the system very useful for Rapid Prototyping (Figures 1.20e and 1.20f).

The last part of the evaluation collected the overall opinion of the participants about the system and its application for Rapid Prototyping in the field of Industrial Design Engineering. The results show that the majority of the participants (except for 5) would use the 3D Scanner to get a mesh of a freshly built model, despite its relatively low quality. Furthermore, the majority agrees that they would prefer to use a more accurate scanner, as expected. Concerning the use of the system for Rapid Prototyping, we get a variety of opinions. Nineteen out of twenty one participants answered this question. Three of them disagreed, seven slightly disagreed, three slightly agreed, five agreed and one strongly agreed. One of the reasons of disagreement was the fact that some people prefer more traditional ways for Rapid Prototyping. Nevertheless, the most prevailing reason of disagreement was the low quality of the mesh that resulted from the scanning procedure. Concerning the communication of ideas, all the participants identified that the system can support this. On the question measuring the usefulness of the system most of them agreed that the system is useful for rapid prototyping. Some of their answers are: “It allows quick adding of information to a 3D object”, “It is nice when you want to test the position of buttons, etc”, “in case only an idea needs to be communicated this might be useful...”, “I think it is very handy to have a virtual form of a prototype”, “you can quickly draw something and save it so you can use the same base for multiple drawings”. Finally, on the statement that the system is enjoyable but not useful, 15 participants disagreed and 5 slightly agreed. Most of the participants underlined the importance of the precision of the scanning result and its effect on drawing. The fact that the scanning procedure yields a low quality virtual model has an effect on the experience of the system. By observation, when the users tried a specific (usually complex) design on an object, and the design appeared jagged or discontinuous because of the geometry and quality of the virtual model, they expressed that they consider it enjoyable, but not in the stages of full application to Rapid Prototyping yet, because of the mesh quality. The full dataset can be found in Appendix D.
Chapter 2: Conclusions and Future Work

This chapter contains the conclusions that were reached from the user evaluation results and the entire work in general. Furthermore, future work and improvements are provided in the second and final section of this chapter.

2.1 Conclusions

This section presents the conclusions that were reached after having completed the present work. Since there are multiple sections which we would like to draw conclusions on, we split the work into five separate categories. Therefore, we will start from the Calibration approach, move on to the Interaction Design, then continue with the 3D Scanner application and finally, we will finish with conclusions on the Interaction with virtual models and the world.

Calibration Approach

As mentioned in Chapter 2 of Part II, the system calibration is achieved when all the hardware components are set with respect to a common reference, the “World” in our case. Due to the fact that one core hardware component of our system comes pre-calibrated, with the only requirement to set a Cartesian coordinate system as reference, we took the decision to use our system’s world as the main reference and calibrate the rest of the hardware accordingly. The first component that we calibrated is the projector. Its performance was more than satisfactory, covering the setting area. The same approach can also be used for a system of multiple projectors, as long as the point correspondences are sampled to cover a relatively large area. Concerning the calibration of the RGB-D camera, its setup was achieved by following two individual (yet complementary) steps. The first one is related to the RGB camera calibration, and the second one to the calibration of the Depth camera. The result of this was acceptable, regarding Kinect’s possibilities. In Section 1.2.1 of Part III we practically tested its accuracy by drawing circles on an array of points of the 2D image coordinates of the RGB camera viewer and projecting spheres on the corresponding locations in the world. The spheres and the circles did not overlap, concluding that its precision is not very high for applications with high accuracy requirements. Nevertheless, the additional approach to transform 3D device coordinates to real world coordinates provides high accuracy results.

Interaction Design

From the user evaluation we had the chance to get valuable feedback on the basic parts of the interaction design directly from the source, the industrial design engineering students. For this specific setting, we believe that the definition of the peripersonal region as the realm for the system’s supported interaction was right. Although this was not directly tested, the feedback on the user input and interface, confirms that tools within the hand region are considered to be more appropriate for interaction with the system. The participants agreed that the constructed pen was appropriate for user input. The fact that designers already use markers and pencils of various hardness for design, renders
the use of a tracked pen simple. Nevertheless, some participants expressed the need to draw in an alternative way, based on the amount of pressure they put on the tracked pen. Some of them claimed that the user input would feel even more natural for them. Regarding the menu as an interface to the system, the results are also in favor of this present design. The majority agreed that using a menu was an appropriate way to provide drawing options to the user. Furthermore, despite the fact that feedback is only provided for color selection, the users found the overall selection of options easy. Several small scale suggestions were made, like changing the color of the selected button for a couple of seconds, in order to provide feedback on the selection. The use of the menu appeared to be easy, and half of the participants believed that there was no need for learning, which means that the design succeeded in achieving its goal of simplicity and intuitiveness. Finally, the visual feedback on the tip of the pen increased user awareness, which is another detail of the design that proved to be useful.

3D Scanner
Apart from the aforementioned, the user evaluation also provided valuable feedback on the Scanner application. We conclude that the cheese platter was a good choice for an assistive surface for scanning. According to all participants, its turning feels natural during the scanning procedure. Moreover, it is a practical way to rotate an object around an axis, which in our case was parallel to the z-axis of the defined world. Nevertheless, the cheese platter, as an assistive surface, has some drawbacks, too. Careful observation revealed that its rotation axis was slightly misplaced, hence, there was a small deviation of the center of rotation. This resulted in a shift of some millimeters of each resulting point cloud, which affects the quality of the scan. Regarding the audio feedback as additional information on the progress of the scanning procedure, we were pleased to see that it was approved by the participants, which establishes it as a successful choice. Furthermore, the scanning process qualified as an easy, quick and low-effort learning procedure. As for the quality of the end result (mesh), despite the fact that some users were satisfied with the result, the majority of users expected a better quality for the virtual model.

Interaction with virtual models and the world
Overall, the binding of the 3D scanner’s result with the tools designed for interaction with virtual models and the world, comprises an immersive and enjoyable experience. The annotation functionality, dynamic objects and the virtual models, encourage the application of the SAR system in the field of Industrial Design Engineering. The functionality of the system has proven to be satisfactory. From the user evaluation, it can be concluded that the system would be useful for Rapid Prototyping. Nevertheless, there is still a need for improvements before its application to the field. The fact that the quality of the virtual model resulting from the scan is low is of major importance regarding the decision of designers to employ it. This allows us to conclude that even though the system’s value is recognized, more effort is needed in order to be actually used as an alternative to the conventional Rapid Prototyping process.

2.2 Future Work
The conclusions have shown that most of the system’s design aspects were successful. Nevertheless, there is always room for improvements and/or changes. From the results, it is confirmed that more effort is needed regarding the 3D scanner. Although Kinect offers a neat solution for its price,
designers underlined that the quality of the mesh is very important. The quality of the resulting mesh of this SAR system is strongly dependent on Kinect. As mentioned in the technical specifications, in Section C.3.2 of Part IV, Kinect’s resolution is 640 x 480. Furthermore, in the accuracy test (Section C.3.3, Part IV), the average error was found to be 1.56cm in the interval of 0.50 to 3.00 meters distance. The average error in the system’s setting, where the distance does not reach more than 1.5 meters, is less than 1cm (around 0.87cm in [0.5m-1.4m]). The accuracy is too low for sampling the points of an object that we might like to reconstruct. Microsoft is now working on the next generation of its RGB-D camera, Kinect 2. It is said that Microsoft aims to design a more accurate product (higher resolution) than the one that is currently on the market. In this case, it would be worth performing similar tests with Kinect 2. Nevertheless, there are other alternatives as well. For example, there are methodologies on improving Kinect’s accuracy, like Patra et al. [12]. Finally, a more promising alternative is to use technology known for its high accuracy, i.e. laser technology. Laser scanners produce very high quality results. Even if the system is not built from scratch, it would be interesting to insert a virtual model produced by a laser scanner, instead of the developed 3D scanner’s result, for the sake of comparison.

Concerning the interaction design for the SAR system, there are a few features that show potential for improvement. Incorporating an extra condition for controlling the drawing state of the current design would be useful. The idea of pressing a button whenever the user needs to draw, would improve the condition of the drawing state (which is currently set to be active in less than 0.5 cm distance). We believe that this would result in a more stable writing and perhaps more natural drawing experience. The same button could also be used in order to add new lines (whenever it is pressed, a line could be inserted). Based on the same concept, another button could be used on the pen to provide the undo command. A simple way to implement these buttons would be to attach two colored stickers on the pen, and use the RGB camera to detect the color on than specific location, which is already known, since the pen is being tracked in real time. Also, real buttons could be used, as long as their size is small, and their shape allows drawing. Another idea that emerged from the user evaluation feedback is to use a pressure sensor on the tip of the pen and adjust the brush size according to the pressure that the user applies. Furthermore, an LED could be attached on the top of the pen (on the 3D crosshair region) in order to display a color (e.g. red) notifying the user that the pen is in drawing state each time the lower pressure threshold is reached. This would limit the aforementioned embedded buttons to one (the undo button).

Regarding the annotation mechanics, the speed of the nearest neighbor search (Part II, Chapter 3) needs to be improved. Right now, this search takes one to two seconds. When drawing on the virtual model, this lag becomes apparent. Of course, this is also related to the number of vertices of the virtual model; the more vertices it has, the higher the latency. A solution for real-time operation needs to be further investigated. Finally, in order for the system to support a variety of geometries and sizes during the surface reconstruction step of the scanning procedure, a useful extension would be a GUI enabling the user to choose the parameters for the reconstruction and make use of the most optimal result.


[Cao 06] Xiang Cao, Ravin Balakrishnan, “Interacting with Dynamically Defined Information Spaces using a Handheld Projector and a Pen”, ACM UIST Symposium on User Interface Software and Technology, October 2006


[Han 06] Sang Heon Han, Jung Hoon Kim, Tae Soo Yun, Dong Hoon Lee, “Extensible Interface using Projector-based Augmentation”, *The 2006 World Congress in Computer Science Computer Engineering, and Applied Computing CGVR*, Las Vegas, Nevada USA (June 26-29, 2006)


[Raskar 01] Ramesh Raskar, Kok-Lim Low, “Interacting with Spatially Augmented Reality”, *AFRIGRAPH*, November 2001


PART IV
APPENDICES
Appendix A: Implementation

This appendix contains information about the platform and the development tools that were used for carrying out this project. Furthermore, details on the communication protocols to connect specific parts of code, due to the use of different programming languages, and libraries, are presented.

Platform, Development Tools and Communication Protocols

The software of this work was developed in two operating systems: Ubuntu 12.04 and Windows 7. The initial idea was to implement a system that is open-source-code friendly. This idea and the fact that the previous system ran in Ubuntu was the reason Ubuntu OS was used as the main development platform. Nevertheless, due to personal use of Windows OS, a large part of software was also tested and sometimes developed on Windows platform as well.

With regard to software libraries, OpenSceneGraph\textsuperscript{25} toolkit was used for the 3D computer graphics. OpenSceneGraph is originally programmed in C++. Nevertheless, a Python wrapper of OpenSceneGraph (osgSWIG\textsuperscript{26}) was used for development, due to its advantage over fast coding and previous affiliation of the TU Delft Graphics Group with this library. For Kinect Calibration the java binding of OpenNI/NITE\textsuperscript{27} library was used. Again in the same sense Java was preferred over C++ for time saving. Note that the OpenNI/NITE wrapper in Python was not complete by that time, so the Java wrapper was used. For finger tracking javaCV\textsuperscript{28} (a Java wrapper for OpenCV\textsuperscript{29}) and the java binding of OpenNI/NITE library were used. Finally, for the 3D scanning application the Point Cloud Library\textsuperscript{30} together with OpenNI was used in C++, since wrappers available. This project was developed with Eclipse Indigo IDE\textsuperscript{31} and its PyDev\textsuperscript{32} plugin for python and Visual Studio 2010 IDE\textsuperscript{33}. For sending information from Kinect to Ubuntu we construct an xmlrpc\textsuperscript{34} server in Java and a python client respectively. Finally, we used VRPN\textsuperscript{35} for getting information from the IR tracker. More specifically, we created a client to receive information from the IR tracker, according to the simple example published on the library’s website\textsuperscript{36}. For more details have a look at the links added as footnotes.

\textsuperscript{25} http://www.openscenegraph.org/projects/osg
\textsuperscript{26} http://code.google.com/p/osgswig/
\textsuperscript{27} http://openni.org/
\textsuperscript{28} http://code.google.com/p/javacv/
\textsuperscript{29} http://opencv.org/
\textsuperscript{30} http://pointclouds.org/
\textsuperscript{31} http://www.eclipse.org/
\textsuperscript{32} http://pydev.org/
\textsuperscript{33} http://www.microsoft.com/visualstudio/eng/team-foundation-service
\textsuperscript{34} http://ws.apache.org/xmlrpc/index.html
\textsuperscript{35} http://www.cs.unc.edu/Research/vrpn/
\textsuperscript{36} http://www.cs.unc.edu/Research/vrpn/vrpn_Tracker_Remote.html
Appendix B: Code Snippets

Sampling of 2D points from Projector Image

As mentioned in Section 2.4.1 of Part II, this is achieved by setting the projection of the viewer to orthogonal/parallel. The viewer is set by the following command in OSG.

```python
text
viewer.getCamera().setProjectionMatrix(osg.Matrixd_ortho(0, 1024, 0, 768, 0, 0))
```

By clicking we get the 2D position of the viewer’s viewport. Nevertheless, we want to draw some feedback on the graphics world, indicating that a specific point is clicked. This is achieved by converting the 2D point from the camera coordinated to 3D graphics world coordinates. The following lines show the way this is achieved.

```python

cam = viewer.getCamera()
vm = cam.getViewMatrix()
pm = cam.getProjectionMatrix()
vpwm = cam.getViewport().computeWindowMatrix()

MVPW = vm * pm * vpwm
invMVPW = osg.Matrixd_inverse(MVPW)

# Conversion from 2D into 3D (it will be a line, therefore we look for near and far points)
nearPoint = invMVPW.postMult(osg.Vec4d(point.x(), point.y(), 0.0, 1.0) )
nearPoint = osg.Vec3d(nearPoint.x(),nearPoint.y(),nearPoint.z())

farPoint = invMVPW.postMult(osg.Vec4d(point.x(), point.y(), 1.0, 1.0) )
farPoint = osg.Vec3d(farPoint.x(),farPoint.y(),farPoint.z())
```

From Computer Vision to Computer Graphics

These two pieces of code were used to insert the calibration results into OSG.

```python

def setFrustum(self, name3D, name2D):

    # The Frustum of the view in CG can be constructed by the intrinsic parameters of a camera, calibrated in CV. Therefore, we get the principle point (u0, v0), the scaling coefficients (au, av), the focal length (f) from the projector calibration, and after setting the image plane size (width, height) according to projector resolution, the frustum in OSG is set.
```

*References*


*Dependencies*
- numpy (python library for scientific computing: numpy.scipy.org)
- osgswig (python wrapper for OpenSceneGraph: code.google.com/p/osgswig)
- name3D: path of the txt file containing the 3D points
- name2D: path of the txt file containing the 2D points
- PythonSVDCalib.py: file containing the projector calibration process

```python
# get calibration instance
calibInstance = PythonSVDCalib.SVDCalibration(name3D, name2D)
# get calibration matrix
calibM = calibInstance.getCalibrationMatrix()
# normalize the calibration matrix
# (divide by the element at 3rd row, 4th column)
calN = calibM/calibM[2][3]

# intrinsics: uo, vo, alpha, beta, f
principlePoint = calibInstance.getImageCenter()
uo = principlePoint[0]
vo = principlePoint[1]
scalingCoefficients = calibInstance.getFocalLength()
alpha = scalingCoefficients[0]
beta = scalingCoefficients[1]

# focal length according to equation 2.104, on page 26,
f = numpy.sqrt((numpy.power(calN[0][0], 2) + numpy.power(calN[0][1], 2)) / (numpy.power(calN[2][0], 2) + numpy.power(calN[2][1], 2) + numpy.power(calN[2][2], 2)) - numpy.power(uo, 2))

# image plane size (projector's resolution)
# also implicitly used for glViewport
width = 1024
height = 768

# Frustum parameters: left, right, bottom, top, near, far
# -----------------------------------------------
near = f  # equals to focal length
far = 10000  # equals to something relatively far

# according to [1], page 52
left = -uo/alpha*near
right = (width-uo)/alpha*near
bottom = -vo/beta*near
top = (height-vo)/beta*near

# osg graphics frustum definition similar to glFrustum
frustum = osg.Matrixd_frustum(left, right , bottom, top, near, far)

return frustum
```
def setModelView(self, name3D, name2D):
    '''
    The ModelView in CG can be constructed by the extrinsic parameters of a camera, calibrated in CV. Therefore, we get the rotation matrix (R) and the translation vector (T) from the projector calibration. Then we fill in a 4x4 matrix with elements of the previous two, and then transpose it to be row-major in compliance with OpenSceneGraph. Furthermore, we multiply the R and the T with -1 so that the position of the projector center falls into the correct place.

*References*
[1] In CV2CG.h the ViewMatrix of the camera: http://code.google.com/p/cv2cg/
    ...

    #get calib instance
calibInstance = PythonSVDCalib.SVDCalibration(name3D, name2D)

    #get translation vector and rotation matrix
    T = calibInstance.getTranslationVector()
    R = calibInstance.getRotationMatrix()

    #transposed and mirrored for OSG acceptance
    modelViewSVDMir = osg.Matrixd(-R[0][0], -R[0][1], -R[0][2], 0,
                                   -R[1][0], -R[1][1], -R[1][2], 0,
                                   -R[2][0], -R[2][1], -R[2][2], 0,
                                   -T[0][0], -T[1][0], -T[2][0], 1)

    return modelViewSVDMir
Appendix C: Hardware Components

This chapter provides an overview of the hardware components of our SAR system. Furthermore, important features and characteristics of each component are being described in this chapter. In some cases we make design decisions which are explicitly written at the end of the respective section.

The main hardware components, included in our SAR system, are a projector, an infrared optical tracker and an RGB-D camera. As mentioned earlier, the projector is the display device and the rest are two different systems for scene tracking. The hardware components are displayed in Figure C.1. In this section, critical technical characteristics of these components are described. Features that play an important role in system’s operation are presented.

![Figure C.1: The main hardware components of this work: LCD projector, optical tracker and RGB-D camera on the left and a DLP projector on the right.]

C.1 Projector

In this section, some basic terminology for this display device is provided. First of all, terms such as projector’s throw distance, throw ratio and focal length are explained. Then, the keystone effect together with steps for correcting it, are described. Finally, this section ends with the design choices for the projector component of our SAR system.
C.1.1 Projector Characteristics

Throw distance, throw ratio and focal length
Throw distance (also known as throw\(^37\)) is the distance between the projector and the display surface, see Figure C.2. Another related term is the throw ratio. The throw ratio refers to the ratio of the throw distance to the projected image width. A larger throw ratio corresponds to a more focused optical system. The throw ratio of a projector can also be calculated from the focal length of the lens \((f)\) and the diagonal size of the LCD/DLP panel according to htrgroup\(^38\).

![Figure C.2: Throw distance and focal length, taken from htrgroup.](http://en.wikipedia.org/wiki/Throw_(projector)

Min and Max throw distance
The minimum (min) throw distance is determined by the lower end of the lens’ focus range. If the throw distance is less than the minimum, the image will be out of focus. The maximum (max) throw distance is usually limited by the brightness of the projector, rather than the upper limit of the focus range. See Figure C.3.

![Figure C.3: Min and Max throw distance, taken from htrgroup.](http://htrgroup.com/main.php?section=all)

If the projector is moved beyond the maximum throw distance, the projected image will be so large that it will not be sufficiently bright. Note that the size of the image, the screen gain and the projector brightness determine the brightness of the projected image, and not the throw distance. The brightness

---

\(^{37}\) http://en.wikipedia.org/wiki/Throw_(projector)

\(^{38}\) http://htrgroup.com/main.php?section=all
is inversely proportional to the area of the screen, i.e. an increase of the screen width will result to brightness decrease.

**Keystone Correction**

Keystone effect occurs when a projector is aligned non-perpendicularly to a screen (directed towards the screen at an angle) or when the projection screen has an angled surface. In such cases, the projected image will appear distorted. The image that results from these misalignments has similar shape to that of a trapezoid; edges furthest away from the projector are wider than those closest to the projector, see left upper image of the bookcase in Figure C.4.

![Figure C.4: Keystone Correction, taken from htrgoup and projectorpeople](http://www.projectorpeople.com)

In order to eliminate this effect, so that the projected image appears squared and not trapezoid, keystone correction is introduced. Keystone correction can take place either manually or digitally. The manual keystone correction is a physical adjustment to the lens of the projector, so that it projects at an angle higher or lower than it would be if it were on a flat surface. This manual adjustment works well in some situations, but is not flexible, is time consuming and needs a lot of effort to get the correct result. On the other hand, LCD and DLP projectors which are both digital display devices offer digital conversion and scaling of the data that is received. The result is a squared image, even at an angle. This allows more flexibility when setting up their projector in variable environments. While keystone correction is a handy feature for portable projector users, it does not produce an ideal picture quality. Scaling inside the projector creates a slightly distorted image, particularly at the sides of the screen. In a long-term installation, lens shift is preferred to keystone correction.

**C.1.2 LCD and DLP Projector Technology**

LCD and DLP projectors are both digital display devices. Each has unique advantages over the other. The technologies and their respective advantages are shortly presented in this section.

---

39 [http://www.projectorpeople.com](http://www.projectorpeople.com)
LCD projectors employ a three-panel LCD (Liquid Crystal Display) system, referred to as 3LCD, according to projectorpeople\(^{40}\). The white light from the projector lamp is split into red, green, and blue beams using two dichroic mirrors, which are special mirrors that only transmit light of a specified wavelength. Each red, green and blue beam then passes through a dedicated LCD panel made up of thousands of miniscule pixels (the higher the resolution, the more pixels). An electrical current turns the panel's pixels on or off to create the grayscale equivalent of that color channel. The three colors are then recombined in a prism and projected through the projector lens and onto the screen. See Figure C.5a, for a visual explanation. LCD projectors reproduce bright, naturally colored images and are also capable of detailed shadow reproduction. LCDs are found in many of today’s electronics (cell phones, mp3 players, etc). Due to the fact that they are thinner, lighter, draw less power than other competing display technologies, LCDs are very common today. Furthermore, 3LCD is the world's most popular projection technology that delivers high quality images for the most demanding tasks.

![Figure C.5: a) LCD projector technology taken from projectorpeople. b) DLP projector technology taken from egadged\(^{41}\).](image)

DLP projectors work quite differently than the previous ones. Instead of having LCD panels through which light is passed, a DLP (Digital Light Processing) chip which is a reflective surface made up of thousands of tiny mirrors, is used to modulate the passing light. Each mirror in the DLP chip represents a single pixel. In a DLP projector, light from the projector's lamp is directed onto the surface of the DLP chip. The mirrors move back and forth, directing light either into the lens path to turn the pixel on, or away from the lens path to turn it off. In order to define color, there is a color wheel consisting of red, green, blue, and white filters. This wheel spins between the lamp and the DLP chip and alternates the color of the light hitting the chip from red to green to blue. The mirrors tilt away from or into the lens path based upon how much of each color is required for each pixel at any given moment in time. This activity modulates the light and produces the image that is projected onto the screen, according to projectorcentral\(^{42}\). See Figure C.5b, for a visual explanation.

**LCD vs DLP**

One benefit of LCD is that it always delivered better color saturation than the corresponding DLP projector. In most single-chip DLP projectors, a white panel is included in the color wheel along with red, green, and blue in order to boost brightness. Though the image is brighter than it would otherwise be, this tends to reduce color saturation, making the DLP picture appear not quite as rich and vibrant. LCD also delivers sharper image than DLP at any given resolution (when a DLP unit is placed side by side with a LCD unit).

\(^{40}\) http://www.projectorpeople.com/lcd-projectors/
\(^{41}\) http://egadged.blogspot.nl/2011/08/perbedaan-dan-pengertian-projector-dlp.html
\(^{42}\) http://www.projectorcentral.com/lcd_dlp.htm
side with an LCD of the same resolution, the LCD typically looks sharper in comparison). A third benefit of LCD is that it is more light-efficient. LCD projectors usually produce significantly higher ANSI lumen outputs than do DLPs with the same wattage lamp.

There are several unique benefits that are derived from DLP technology. One of the most obvious is small package size, a feature most relevant in the mobile presentation market. Since the DLP light engine consists of a single chip rather than three LCD panels, DLP projectors tend to be more compact. All of the current pico-projectors on the market are DLPs. Most LCD projectors are from two and a half kilos and up. Another DLP advantage is that it can produce higher contrast video with deeper black levels than you normally get on an LCD projector. DLP has ardent followers in the home theater world primarily due to this key advantage. A third competitive advantage of DLP over LCD is reduced pixilation (screendoor effect). These days it is most relevant in the low priced, low resolution SVGA class of products. In SVGA resolution, DLP projectors have a muted pixel structure when viewed from a typical viewing distance. Conversely, most SVGA-resolution LCD projectors tend to have a more visible pixel grid. While this is entirely irrelevant in slide presentations, it becomes problematic in video presentations. For this reason, SVGA-resolution LCD projectors are not recommended for home theater use. In XGA and higher resolution, DLP technology pretty much eliminates pixel visibility from a normal viewing distance. However, in the latest WXGA resolution LCDs do so as well. So with higher resolutions, differences in pixelation are not the big competitive battleground they used to be. DLP continues to hold a small competitive edge, but the dramatic advantage of DLP over LCD no longer exists.

C.1.2 Pinhole Camera Model for Projector

A projector is often characterized as the “inverse” of a camera. It has the same parameters as the camera, and the difference lies in the fact that light flows in reverse (projection onto scene objects). Since the camera and the projector share the same parameters, the projector adheres to Pinhole Camera Model as well. Therefore, the same rules and approaches can be applied. More information about pinhole camera model, its geometrical and mathematical background is provided in Part II, Section 2.1.1 and Section 2.1.2.

C.1.3 Design choices

After having provided comprehensive details about digital projector technology, we now define the important attributes and characteristics of the projector technology used in this project. As the setup environment of the system is not static, but can change each time it is being set, projector characteristics such as throw distance, focal length, min and max throw distance are empirically set each time. This means that during the system setup, the position of the projector is set based on the configuration of the environment at the optimal position based on experimentation and observation. The same applies for the setting of the focal length. In this case, the lens is adjusted, so that the projected image is as clear as possible. Regarding the keystone correction, we decided not to use it, simply because the shape of the projected image is indifferent for us. Nevertheless, it is important that the shape stays the same, since the projector calibration will be based on that initial shape. One important attribute is that we make use of the pinhole camera model for the projector. More details are given in Chapter 2 of Part II on that. The system is formally designed to work with an LCD projector, nevertheless a DLP projector is also used for a small technical comparison and to prove that the
software works equally well with both technologies. The following table summarizes the decisions taken on the design, based on the afore-mentioned.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throw distance, throw ratio and focal length</td>
<td>Empirically set</td>
</tr>
<tr>
<td>Min and Max throw distance</td>
<td>Empirically set</td>
</tr>
<tr>
<td>Keystone Correction</td>
<td>Not Used</td>
</tr>
<tr>
<td>Pinhole Camera Model for Projector</td>
<td>Used</td>
</tr>
<tr>
<td><em>LCD technology</em></td>
<td>Used</td>
</tr>
<tr>
<td><em>DLP technology</em></td>
<td>Used for comparison reasons</td>
</tr>
</tbody>
</table>

Table C.1: Summary of design choices for projector technology.

C.2 Tracker

This section describes the second hardware component of our SAR system, the tracker. The first subsection provides a description of the optical tracking operation, and finally the specifications of the used infrared optical tracker are given, in order to have an overview of the system we are using, its advantages and possible disadvantages. Since the tracker and its SDK arrive fixed, there is no space for design choices in this section. The use of the specific tracker acts as a design choice.

C.2.1 Optical tracking

For tracking and registration of objects in the scene, optical tracking is one of the enabling technologies appearing in this work. More specifically, an optical tracker which consists of an optical imaging system, a mechanical platform and a processing unit, is used. The optical imaging system, in our case, is two infrared cameras. Infrared light is emitted to the scene via a transmitter, and retro-reflective markers are used for reflecting the IR light back to the receivers, which in this case are the infrared cameras. These two cameras offer two different viewpoints and through triangulation the position of the marker can be retrieved. IR illumination works satisfactorily in indoor environments, therefore this type of tracking is confined within this type of environments (indoor tracking). The retro-reflective markers can be placed onto objects (Figure C.6) and form a model of the object, also known as *device model*. Usually special software comes with the tracker, which is responsible for training and tracking the device models. Figure C.7 illustrates the PS-tech infrared optical tracker together with a laptop running the accompanied software.
C.2.2 Optical tracker specifications

The Personal Space Technologies tracker was used as infrared optical tracker, also called PS-tech tracker. The specifications of the specific machine are provided below:

- Refresh rate at 55 Hz
- Precision measured with 7mm markers at a distance of 60 cm <1mm, <1deg
- Minimum latency at 18 ms
- Pre-calibrated unit
- Origin definition is by setting the reference point with only one click
- 6 DOF with 4 or more marker, 3 DOF with a single marker
- Supports passive markers (retro-reflective stickers and spheres) and active markers (led)
- Hardware interface through Ethernet
- Software interface through VRPN and via SDK
- Output in x, y, z positional coordinates and orientation angles, rotation matrix or quaternions

These specifications are important to take into account for better understanding of the overall system capabilities.

43 http://ps-tech.com/tracking/pst-55/
C.3 RGB-D Camera

This section describes the third and final hardware component of our SAR system, the RGB-D camera or else Kinect. The first sub-section provides a description of the notions of 2D and 3D cameras and Kinect. Later on, the critical features of Kinect RGB-Depth camera are provided, which are actually the hardware specifications. In the third subsection a small test about Kinect accuracy is provided and the final section ends with another test about infrared interference in our hardware setup. Again the decision to use Kinect acts as a design choice.

C.3.1 2D Camera, 3D Camera and Kinect

Kinect is an accessory of Microsoft Xbox 360 game console. It was released as a “controller-free gaming and entertainment experience”, supporting interaction with the system without using any mark or controller. It consists of a depth-sensing camera which uses structured light (IR projector and IR camera), a color (RGB) camera, an array of microphones, and a motorized pivot for tilting the sensor. Figure C.8 illustrates Kinect Xbox 360 device. By combining the depth and the color image sequences, the real 3D objects found in device’s field of view (FOV), can be virtually recreated at their correct sizes and be projected back into 3D space.

![RGB-D Kinect Camera](http://www.xbox.com)

According to Kreylos’ notes\(^{45}\), any conventional camera works by projecting a 3D scene (collection of 3D points) onto their 2D image plane. Projection lines connect the 3D points with their 2D projections passing through camera’s lens. In conventional 2D cameras, once the 3D points are projected onto the camera’s image plane, there is no way back in retrieving their 3D coordinates and being able to reconstruct these points. In other words, the distance that the 3D point has crossed along its projection line is lost.

On the other hand, 3D camera like Kinect provides this missing bit of information for 3D reconstruction. For each 2D pixel on the image plane, instead of the color information, also the 3D point’s distance along its projection line is given. Note, that the scene 3D reconstruction can take place only by the side of the viewing direction of Kinect. Multiple Kinect devices need to be used to reconstruct entire (complete) 3D objects simultaneously.

Usually stereoscopic cameras are referred to as 3D cameras. However this is a bit misleading since these cameras have two different lenses and capture instances from two different viewpoints, which are carefully chosen to create the effect of 3D in viewers’ brains. On the other hand, real 3D pictures

\(^{44}\) [http://www.xbox.com](http://www.xbox.com)

can be viewed from any viewpoint. The whole process simply requires rendering of the reconstructed 3D objects from a different perspective.

C.3.2 The RGB-Depth Kinect Camera

Kinect uses a right-handed coordinate system, as shown in Figure C.9 in red axes. The position of the IR projector, IR camera and RGB camera within the device are displayed in the same figure, taken from [Smisek 11]. In this figure, if \( X1 \) is a point in 3D, the projections lines connecting it with the center of each device element respectively, are displayed in blue. The IR camera is usually aligned to the RGB camera. The perspective features of the RGB camera and the Depth sensor, are provided in the following subsections.

**Kinect RGB Camera Features**
- 8-bit VGA resolution
- 640x480 pixels
- captures at a frame rate of 30 Hz
- angular field of view of 57° horizontally and 43° vertically

**Kinect Depth sensor Features**
- infrared (IR) projector and infrared (IR) camera
- infrared camera of 640x480 pixel resolution with 11-bit depth (\( 2^{11} = 2048 \) levels of sensitivity)
- practical ranging limit from 1.2m to 3.5m with Xbox software

The measurement of the depth is described as a triangulation process. The IR source emits a single beam which is then split into multiple beams to create a constant pattern of speckles projected onto the scene. This pattern is then captured by the IR camera and is being compared with a reference.
The Kinect device captures depth and color images simultaneously at a frame rate of 30 fps. The integration of depth and color results into a point cloud of 307,200 points (640 x 480) in each frame. The registration of consecutive images increases the point density. This contributes to a complete point cloud of an indoor scene in real time.

**C.3.3 Accuracy of RGB-Depth Kinect Camera**

In order to ascertain the linear relationship between the measurements taken by Kinect and the real values, we decided to carry out a small test. As Figure C.10a shows, we placed a piece of duct tape on the floor, measured it in centimeters and then placed Kinect at the start of the tape (0 cm). In the same figure on the upper left corner a wooden object is highlighted. This object is a rectangular parallelepiped or else cuboid and is used to measure the distance. A point on the front face of the object (the green one) is defined as a reference point for the experiment. Note that Kinect returns as a depth estimate the distance given by the perpendicular line (blue in Figure C.10b) to the front end of the device, and not by the actual distance of the object to the lens (red in Figure C.10b).

![Figure C.10: a) Setup of the accuracy test b) Kinect depth value measurement, taken from [Andersen 11].](image)

Figure C.11: View of Kinect RGB camera after the measuring process.
The rest of the process includes placing the measurement object in variable lengths and clicking on the green point found on the object at the RGB camera view window. The two views of the Kinect cameras, RGB and Depth, are aligned (functionality offered by OpenNI software). Nevertheless, we only care for the Z-axis measurement result given by Kinect, which is also the same in not aligned views. After measuring from approximately 0cm to 300cm with a step of 10 cm, we get 30 measurements in total. An instance of the RGB camera view of Kinect, after all the measurements were taken by clicking in 2D and measuring in 3D, is illustrated in Figure C.11. Note that for the first 40cm Kinect is unable to provide a depth measurement, see red cells in Table C.2.

<table>
<thead>
<tr>
<th>Kinect Depth (cm)</th>
<th>Real Distance (cm)</th>
<th>Abs Diff (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>0</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>0</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>61.2</td>
<td>62</td>
<td>0.8</td>
</tr>
<tr>
<td>71.1</td>
<td>72</td>
<td>0.9</td>
</tr>
<tr>
<td>81.4</td>
<td>82</td>
<td>0.6</td>
</tr>
<tr>
<td>91.3</td>
<td>92</td>
<td>0.7</td>
</tr>
<tr>
<td>101.1</td>
<td>102</td>
<td>0.9</td>
</tr>
<tr>
<td>110.9</td>
<td>112</td>
<td>1.1</td>
</tr>
<tr>
<td>121.4</td>
<td>122</td>
<td>0.6</td>
</tr>
<tr>
<td>130.6</td>
<td>132</td>
<td>1.4</td>
</tr>
<tr>
<td>141.3</td>
<td>142</td>
<td>0.7</td>
</tr>
<tr>
<td>151.2</td>
<td>152</td>
<td>0.8</td>
</tr>
<tr>
<td>161.8</td>
<td>162</td>
<td>0.2</td>
</tr>
<tr>
<td>172.3</td>
<td>172</td>
<td>0.3</td>
</tr>
<tr>
<td>182.4</td>
<td>182</td>
<td>0.4</td>
</tr>
<tr>
<td>192.5</td>
<td>192</td>
<td>0.5</td>
</tr>
<tr>
<td>203.9</td>
<td>202</td>
<td>1.9</td>
</tr>
<tr>
<td>214</td>
<td>212</td>
<td>2</td>
</tr>
<tr>
<td>225.2</td>
<td>222</td>
<td>3.2</td>
</tr>
<tr>
<td>234.3</td>
<td>232</td>
<td>2.3</td>
</tr>
<tr>
<td>246</td>
<td>242</td>
<td>4</td>
</tr>
<tr>
<td>253.2</td>
<td>252</td>
<td>1.2</td>
</tr>
<tr>
<td>264.9</td>
<td>262</td>
<td>2.9</td>
</tr>
<tr>
<td>275.4</td>
<td>272</td>
<td>3.4</td>
</tr>
<tr>
<td>284.5</td>
<td>282</td>
<td>2.5</td>
</tr>
<tr>
<td>296.7</td>
<td>292</td>
<td>4.7</td>
</tr>
</tbody>
</table>

Table C.2: Testing Kinect accuracy: real vs measured distance.

Average Abs Difference: **1.56**
As Table C.2 shows, the average absolute difference between measured and real distance is 1.56 cm. Putting all 30 values in a diagram, we conclude that the relationship between real and measured distance is linear, see Figure C.12. Moreover, the total average difference is relatively low (1.5 cm), and this allows us to conclude that there shouldn’t be a problem in the future measurements with Kinect. Note that the difference starts to increase from 2m and on. This means that the accuracy decreases as the object moves farther away from Kinect device. For a more detailed overview of Kinect accuracy read [Khoshelham 12] and [Andersen 11].

![Kinect depth vs real distance measurements](image)

Figure C.12: Linear relationship between measured by Kinect and real distance.

C.3.4 Infrared Interference

There have been several tracking problems reported due to infrared interference while using multiple Kinect devices, see [Viager 11]. More specifically, interference between overlapping structured light patterns from multiple Kinect cameras pointing at the same area produces invalid and noisy depth pixels. Solutions to overcome this problem have been provided by works such as [Butler 12] and [Maimone 12], proposing the vibration of each Kinect independently which introduces motion blur of the structured light patterns from other devices. See Figure C.13 and Figure C.14 for more information.

![Interference between overlapping structured light patterns](image)

Figure C.13: a) Interference between overlapping structured light patterns from two Kinect cameras, b) Result after shake ‘n’ sense method, c) Significant artifacts before the method, d) Point cloud after the method, all taken from [Butler 12].
Although our system does not use multiple Kinect devices, but just only one, and doesn’t face the risk of overlapping structured light patterns, the fact that there is another source of infrared lighting in the setup, the tracker, makes the testing of possible interference important during these first stages of design. After taking the reported interference instances into account, we find out if our system faces similar problems by carrying out two simple tests.

In the first test, after turning on the infrared tracker, we get instances from Kinect depth image view to examine any weird behavior, or any noisy output. For that we hold on a card with retro-reflective sticker that is currently being tracked by the tracker, as Figure C.15a shows. The result implies that there is no noticeable effect while carrying out this small test, thus we conclude that it seems that there are no infrared interference problems, see Figure C.15b.

To double check the previous result, we perform another small test. This time we view the environment via the two infrared cameras of the tracker as shown in the figure below. At first we take a screenshot while Kinect is turned off, then a screenshot while it’s on and finally we subtract these two images to examine if any weird behavior or noise is observed. As the last image of Figure C.16 and the previous test imply, there are no interference problems between these two hardware components of our system.
Figure C.16: Simple test to check the existence of possible infrared interference from the infrared optical tracker camera views. Image a) taken with RGB-D camera off, image b) taken with RGB-D camera on (speckles are visible), and figure c) is the difference image between a) and b).
# Appendix D: User Evaluation Material

## Pre-test Questionnaire

| Name:.......................................................................................................................... |
| Age:.................. Gender: M / F Study:.................................................................. |

Please check the box that corresponds to your answer

<table>
<thead>
<tr>
<th>Familiarity with the field of Spatial / Projected Augmented Reality</th>
<th>None</th>
<th>Little</th>
<th>Medium</th>
<th>Good</th>
<th>Excellent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Familiarity with using 3D Scanners</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Familiarity with Rapid Prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Familiarity with our system (used it at INSYGHTLab Opening or any other time)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PART A: *Scan an object*

**Task:** scanning of a simple and/or complex physical object (built prototype) to get the polygonal mesh of the object. The process is described in steps by the figure below.

![Scan an object figure](image)

PART B: *Interaction with the model*

**Task1:** Draw one star shape on the static cube and one on the scanned cube like the following figure shows.

![Interaction with the model figure](image)
Task 2: Keep a note on the surface of the table like the figure shows.

Task 3: a) Draw a simple shape such as a square, a triangle or a circle on one of the object faces by using color(s) and brush size of your choice

b) Draw a more complex shape on the object by using color and brush size of your choice

c) Repeat for the complex object scan

Task 4: Save the scene. View the changes.
The present questionnaire is part of the user evaluation process of the system that was designed and developed for the purposes of a master’s thesis project. This questionnaire is handed to you after having completed a number of tasks by using the system.

The questionnaire consists of four sections: (A) Scanner Application, (B) Interaction with the scanned model, (C) SAR system, and (D) Remarks. The aim is to gather data with respect to system’s Usability, Functionality, Usefulness and User Experience. Your data will be used for evaluating the present SAR system and will be fully protected. In case you have any questions contact the experiment supervisor.

Below you can find some guidance on filling in this questionnaire. Thank you in advance for taking the time to provide us with your input.

**Guidelines:**

- Fill in this questionnaire by placing a mark (e.g. X) in the box that represents your answer
- The N/A option stands for “Not Applicable”
- In section D (Remarks), please write down the general impression that you have (in case you haven’t any remarks)

<table>
<thead>
<tr>
<th>A. Scanner Application</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Audio was a successful way to provide feedback about the progress of the scanning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 The turning of the cheese platter felt natural during the interaction with the scanner</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3 I found the scanning process difficult to carry out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In case you agree explain why:</td>
</tr>
<tr>
<td>1.4 The operation of the scanner was intuitive and didn’t require learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5 I am satisfied with the resulting mesh for use during rapid prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6 The scanning process is quick</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7 There were no technical difficulties during the scanner operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### B. SAR Application

#### Task 1

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>It was much easier to draw onto the static cube than the dynamic one</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In case you agree explain why:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>The precision of drawing was pretty much equal in both cases (static cube, dynamic cube)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>I preferred to draw on the dynamic cube due to its mobility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Designing on an object that can be moved in space makes me gain more perspective on the future design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>During rapid prototyping accuracy is not an issue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In case you disagree explain why:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Task 2

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Drawing on the table’s surface was easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Being able to keep notes in such a setting is useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Task 3

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Drawing simple shapes on the simple object was easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In case you disagree explain why:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>Drawing simple shapes on the complex object was easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Drawing complex shapes on the simple object was difficult</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Drawing complex shapes on the complex object was easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In case you disagree explain why:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B. SAR Application</td>
<td>Task 4</td>
<td>Strongly Disagree</td>
<td>Disagree</td>
<td>Slightly Disagree</td>
<td>Slightly Agree</td>
<td>Agree</td>
<td>Strongly Agree</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>--------------------</td>
<td>----------</td>
<td>-------------------</td>
<td>---------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>4.1</td>
<td>Whenever I was not satisfied with my drawing the clear button allowed me to redo my design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>The save button is very important because it saves my design to a file which I can view and use later</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>The save button enables me to send my design to my collaborators to get a draft idea on the prototype I am working on</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. SAR Application</th>
<th>User Interface and Tools</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>The feedback on the tip of the pen increased my awareness</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>The use of the pen for user input is an appropriate approach for a system like this</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>The movability of the menu was convenient for me</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>The clicking on the menu buttons was satisfactory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>The color selection was easy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>The brush size selection was difficult</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>The adding of a new line was a bit overwhelming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In case you agree explain why:</td>
</tr>
<tr>
<td>5.8</td>
<td>A spatial menu is an appropriate way to interface with a system like this for rapid prototyping purposes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In case you agree explain why:</td>
</tr>
<tr>
<td>5.9</td>
<td>The lack of physical buttons did not affect my interaction with the menu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.10</td>
<td>The menu was easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.11</td>
<td>The menu required some learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### B. SAR Application Functionality

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Color quality was good</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>The system was quick to detect which buttons I was trying to press</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>The tracking of the object/menu/pen felt real-time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>There was no lag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In case you disagree explain why:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td>The graphics were displayed at the correct position</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### C. SAR System Usefulness

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Slightly Agree</th>
<th>Agree</th>
<th>Strongly Agree</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>I would use this scanner application to get a mesh of a freshly built model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>Precision matters to me during rapid prototyping, so I would use a scanner of better quality like laser technology to get the mesh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In case you agree explain why:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>I would use the SAR system in order to perform rapid prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In case you disagree explain why:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4</td>
<td>I would expect more options such as textures, light, shaders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>The system has the potential to support the communication of ideas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>I consider a system like this very useful for Rapid Prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*In case you agree explain why:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.7</td>
<td>I consider a system like this enjoyable but not useful</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.8</td>
<td>Using a system like this enables me to envision the object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.9</td>
<td>Compared to the traditional way of rapid prototyping, the system makes Rapid Prototyping more sustainable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.10</td>
<td>Compared to the traditional way of rapid prototyping, the system makes Rapid Prototyping more time saving</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remarks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td>Age</td>
<td>Gender</td>
<td>Familiarity with SAR</td>
<td>Familiarity with 3D Scanners</td>
<td>Familiarity with Rapid Prototyping</td>
<td>Familiarity with User System</td>
<td>A_Scanner_Q1</td>
</tr>
<tr>
<td>---------</td>
<td>-----</td>
<td>--------</td>
<td>---------------------</td>
<td>--------------------------</td>
<td>----------------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>21</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>23</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>22</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>23</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>22</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>21</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>20</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>20</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>25</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>20</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>20</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>24</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>22</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>21</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>23</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>22</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>23</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A_Scanner_Q6</th>
<th>A_Scanner_Q7</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
<th>B_SAR_App_Tas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>B_SAR_App_Tas_k_3_Q4</td>
<td>B_SAR_App_Tas_k_4_Q1</td>
<td>B_SAR_App_Tas_k_4_Q2</td>
<td>B_SAR_App_Tas_k_4_Q3</td>
<td>B_SAR_App_Ul_oools_Q1</td>
<td>B_SAR_App_Ul_oools_Q2</td>
<td>B_SAR_App_Ul_oools_Q3</td>
<td>B_SAR_App_Ul_oools_Q4</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>C_SAR_system_</td>
<td>C_SAR_system_</td>
<td>C_SAR_system_</td>
<td>C_SAR_system_</td>
<td>C_SAR_system_</td>
<td>A_Scanner_REV</td>
<td>B_SAR_App_Tes</td>
<td>B_SAR_App_UL</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>Usefulness_Q5</td>
<td>Usefulness_Q6</td>
<td>Usefulness_Q7</td>
<td>Usefulness_Q8</td>
<td>Usefulness_Q9</td>
<td>Q3</td>
<td>i_3_REV_Q3</td>
<td>cols_REV_Q6</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>