

Interaction in Dense One-Handed Handheld Augmented Reality

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Benjamin Venditti

Matrikelnummer 0927121

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Mitwirkung: Dr. Annette Mossel

Wien, 01.11.2014

(Unterschrift Verfasser)

(Unterschrift Betreuung)

Interaction in Dense One-Handed Handheld Augmented Reality

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Media Informatics

by

Benjamin Venditti

Registration Number 0927121

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Assistance: Dr. Annette Mossel

Vienna, 01.11.2014

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Benjamin Venditti
Kohlgasse 38/21, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)

(Unterschrift Verfasser)

Acknowledgements

Optional acknowledgements may be inserted here.

Abstract

The rapid improvement of handheld hardware devices enable real-time rendering of a large number of 3D models in handheld augmented reality (AR) environments. Although there are already many AR applications available for handheld devices, they often remain limited regarding their degree of interactivity compared to desktop AR applications. Important interactions in AR environments are the effortless creation of 3D models, precise selection and six-degree-of-freedom (6DOF) manipulation of virtual objects. Due to the imprecise input of the touch interface, the small screen size and complex touch gestures, all three interactions suffer in one-handed handheld AR setups of different limitations. In this thesis novel one-handed handheld interaction techniques for modeling, selection and manipulation of virtual content are introduced. A simple polygonal modeling technique is presented that uses real-time shape detection to collect hand-drawn shapes that are used for *Perspective Driven Modeling* by extrusion or lathing in an AR environment. The novel selection technique *DrillSample*, designed particularly for the precise selection and disambiguation of virtual objects, is evaluated against state of the selection techniques. *DrillSample* is inspired of taking a core sample, e.g. of earth sediments, and designed with a focus on simple touch input and virtual context preservation. Two new and competing manipulation techniques, *3DTouch* and *HOMER-S*, for the intuitive translation, rotation and scaling of virtual objects are introduced. *3DTouch* focuses on simple touch input and degree of freedom separation, down to a single DOF, making the different DOF easily accessible from different view perspectives. *HOMER-S* on the other hand provides integral manipulation up to 6DOF and avoids touch input completely by mapping the handheld's device pose to the manipulated object. Both techniques are evaluated in a thorough user study followed by a statistical evaluation.

Kurzfassung

Die rasch fortschreitende Entwicklung mobiler Endgeräte erlaubt die Echtzeit Darstellung von vielen 3D Modellen in mobilen Augmented Reality (AR) Anwendungen. Obwohl bereits viele AR Anwendungen für mobile Endgeräte verfügbar sind, bleiben diese in Bezug auf dessen Interaktivität, verglichen mit Desktop AR Anwendungen, eingeschränkt. Wichtige Interaktionen in AR Umgebungen sind die einfache Erstellung von 3D Modellen, präzise Selektion sowie die Manipulation in 6 Freiheitsgraden (6DOF) von virtuellen Objekten. Aufgrund der ungenauen Eingabe des Berührungsbildschirms, der geringen Bildschirmgröße und komplexer Berührungsgesten sind alle drei Interaktionen, in einhändigen Anwendungen mobiler AR, Einschränkungen unterworfen. In dieser Arbeit werden neue einhändige Interaktionstechniken (IT) zur Modellierung, Selektion und Manipulation von virtuellen Inhalten präsentiert. Eine IT zur einfachen polygonalen Modellierung wird vorgestellt. Die Interaktionstechnik erlaubt die Echtzeit Formfassung von Hand gezeichneter Formen, um diese anschließend durch *Perspective Driven Modeling* mittels Extrusion oder Drehung in einer AR Umgebung als Körper zu modellieren. Die neue Selektionstechnik *DrillSample*, speziell entwickelt zur präzisen Auswahl und Unterscheidung von virtuellen Objekten, wird präsentiert und gegenüber modernen Selektionstechniken bewertet. Beim Entwurf von *DrillSample* standen die Bedienung durch einfache Berührungsgesten sowie der Erhalt des virtuellen Kontextes einer Mehrfachselektion, zum Zweck der Objektunterscheidung, im Vordergrund. Zwei neue und konkurrierende Manipulationstechniken, *3DTouch* und *HOMER-S*, zur intuitiven Translation, Rotation und Skalierung von virtuellen Objekten werden vorgestellt. Das Hauptaugenmerk bei der Entwicklung von *3DTouch* liegt auf der einfachen Bedienung mittels Berührungseingabe und der Reduktion der gleichzeitig manipulierbaren Freiheitsgrade. Je nach gewählter Betrachtungsperspektive zum manipulierenden Objekt, sind unterschiedliche Freiheitsgrade zugänglich. *HOMER-S* ermöglicht hingegen die gleichzeitige Manipulation von bis zu 6 Freiheitsgraden und vermeidet die Berührungseingabe gänzlich, indem die räumliche Position und die Orientierung des mobilen Endgerätes auf das zu manipulierende Objekt abgebildet werden. Beide Techniken wurden in einer detaillierten Benutzerstudie und anschließender statistischer Auswertung untersucht.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Problem Statement | 1 |
| 1.3 | Aim of the Work | 2 |
| 1.4 | Individual Publications | 3 |
| 1.5 | Outline | 3 |
| 2 | Related Work | 5 |
| 2.1 | Augmented Reality | 5 |
| 2.1.1 | Definition & Characteristics | 5 |
| 2.1.2 | Augmented Reality Interaction | 6 |
| 2.1.3 | Handheld Augmented Reality | 7 |
| 2.2 | Polygonal Modeling | 8 |
| 2.2.1 | Basic Modeling Operations | 9 |
| 2.2.2 | Handheld Modeling Applications | 9 |
| 2.3 | Selection | 12 |
| 2.3.1 | Ray-casting | 12 |
| 2.3.2 | Handheld Ray-casting adaptations | 12 |
| 2.3.3 | SQUAD | 14 |
| 2.3.4 | EXPAND | 15 |
| 2.4 | Manipulation | 16 |
| 2.4.1 | Virtual Hand | 16 |
| 2.4.2 | Z-Technique | 18 |
| 2.4.3 | Dual-Finger mobile 3D interaction techniques | 19 |
| 2.5 | Touch Gestures | 21 |
| 3 | Methodology | 23 |
| 3.1 | Requirements | 23 |
| 3.2 | General Guidelines | 24 |
| 3.3 | Modeling | 24 |
| 3.3.1 | Guidelines | 24 |
| 3.3.2 | Prefab Based Modeling | 24 |
| 3.3.2.1 | Capturing Shapes | 26 |

| | | |
|----------|--|-----------|
| 3.3.2.2 | Perspective Driven Modeling | 26 |
| 3.4 | Selection | 29 |
| 3.4.1 | Guidelines | 29 |
| 3.4.2 | State-of-the-Art Selection Techniques | 29 |
| 3.4.2.1 | <i>Ray-casting</i> | 30 |
| 3.4.2.2 | <i>Expand</i> | 30 |
| 3.4.3 | DrillSample Selection Technique | 31 |
| 3.4.3.1 | Algorithm | 32 |
| 3.4.3.2 | Important Aspects | 34 |
| 3.5 | Manipulation | 37 |
| 3.5.1 | Guidelines | 37 |
| 3.5.2 | State-of-the-Art Selection Manipulation Techniques | 37 |
| 3.5.3 | 3D Touch | 37 |
| 3.5.3.1 | Degree of Freedom Limitation | 38 |
| 3.5.3.2 | Translation | 40 |
| 3.5.3.3 | Rotation | 40 |
| 3.5.3.4 | Scaling | 41 |
| 3.5.4 | HOMER-S | 42 |
| 3.5.5 | Mode Switches | 43 |
| 3.5.6 | Important Aspects | 44 |
| 4 | Implementation | 45 |
| 4.1 | System Design | 45 |
| 4.1.1 | Hardware | 45 |
| 4.1.2 | System Overview | 46 |
| 4.1.3 | Applications | 46 |
| 4.1.4 | Frameworks | 47 |
| 4.1.5 | Tasks & Overview | 48 |
| 4.2 | Modeling | 49 |
| 4.2.1 | Shape Detection User Interface | 49 |
| 4.2.2 | Image Processing | 50 |
| 4.2.3 | Modeling User Interface | 53 |
| 4.3 | Content Distribution | 53 |
| 4.3.1 | Data Transmission Analysis | 54 |
| 4.3.2 | Networking in Unity | 55 |
| 4.3.3 | Distribution Procedure | 57 |
| 4.4 | Inter Process Communication | 59 |
| 4.4.1 | IPC Mechanisms | 59 |
| 4.4.2 | IPC Architecture | 60 |
| 4.4.2.1 | JSON Remote Method Invocation | 61 |
| 4.4.2.2 | Storage Service | 62 |
| 4.4.2.3 | Storage Client | 63 |
| 4.5 | Selection & Manipulation | 64 |

| | | |
|----------|------------------------------------|------------|
| 4.5.1 | Composite Interaction | 65 |
| 4.5.2 | Selection Interface | 66 |
| 4.5.3 | Manipulation Interface | 67 |
| 4.6 | UserStudy Application | 69 |
| 5 | Evaluation & Discussion | 73 |
| 5.1 | Shape Detection | 73 |
| 5.1.1 | Per Pixel Operations | 73 |
| 5.1.2 | Per Point Operations | 74 |
| 5.1.3 | Detection Robustness | 75 |
| 5.2 | Modeling | 77 |
| 5.3 | Experimental User Study | 79 |
| 5.3.1 | Design & Procedure | 79 |
| 5.3.2 | Subjects and Apparatus | 80 |
| 5.3.3 | Statistical Foundation | 80 |
| 5.4 | Selection | 82 |
| 5.4.1 | Design & Objectives | 82 |
| 5.4.2 | Test Scenarios | 83 |
| 5.4.3 | Results | 84 |
| 5.4.3.1 | Quantitative Evaluation | 85 |
| 5.4.3.2 | Subjective Evaluation | 87 |
| 5.4.4 | Qualitative Evaluation | 88 |
| 5.5 | Manipulation | 89 |
| 5.5.1 | Design & Objectives | 89 |
| 5.5.2 | Test Scenarios | 89 |
| 5.5.3 | Results | 92 |
| 5.5.3.1 | Performative Evaluation | 93 |
| 5.5.3.2 | Subjective Evaluation | 94 |
| 6 | Conclusion & Outlook | 97 |
| 6.1 | Modeling | 97 |
| 6.2 | Selection | 98 |
| 6.2.1 | Outlook | 100 |
| 6.3 | Manipulation | 100 |
| 6.3.1 | Outlook | 102 |
| 7 | Appendix | 103 |
| | Bibliography | 117 |

Introduction

1.1 Motivation

Augmented Reality (AR) is the real-time augmentation of a human sensory perception, such as seeing and hearing, with simulated information generated by a computer. A key functionality of AR is the registration of the real world within the virtual world so that virtual information can be displayed locally and perspective-correctly. AR has proven to be useful in a wide area of applications like psychological treatment of phobias, medical education, edutainment as well as game development. It offers intuitive ways of interaction in artificial 3D environments by employing implicit human knowledge about 3D worlds and naturally trained skills. Due to expensive hardware and impractical gear involved, such as attached wires for the power supply, only few or simple augmented reality applications have entered the mass market. However, recently emerged handheld devices enable real-time rendering of a large number of 3D models in handheld AR environments. Its further proliferation could mark a turning point and specially applications in educational institutions could benefit from the high availability and the “attractiveness” of the new experience that AR provides.

As AR aims to supplement our perception of reality with virtual information, it has to provide ways to interact with virtual objects. Fundamental real-world interactions are the creation, selection and manipulation of objects and are therefore fundamental interactions for virtual objects in AR environments as well. However, when using a handheld device only one hand can be used for input and the available input space on the touchscreen is very limited. All three interactions suffer in one-handed handheld AR environments of different limitations.

1.2 Problem Statement

Existing modeling applications for handhelds make excessive use of the touch interface and provide either simple prefab-based (e.g. cubes or spheres) modeling or desktop-like functionality. Prefab-based modeling is fast and easy, but offers little flexibility for creating arbitrary shaped

objects. Desktop-like modeling applications offer powerful tools to create detailed models, but require a considerable knowledge to be used properly. In both applications the handheld itself is only used as a display with a touch interface, but instead it could be used as a tool for modeling itself, once it is registered in an AR environment.

With increasing computational power, handheld AR environments will be modeled with increasing detail to provide higher levels of realism. In such dense environments selection may become infeasible for objects that are partly or completely occluded or highly similar to surrounding virtual scene objects. A single touch on the device's screen is likely to overlap multiple objects in a dense environment. A state-of-the-art approach is to immediately select the closest object. However, immediate selection is cumbersome in dense environments, as occluded objects need to be moved to access hidden ones. Another approach presents all selected objects, arranged on a grid or a list, and enables the user to refine it's selection in a second step. However, choosing one of many objects from a list leads to unintentional false selections, in case the objects are highly similar in visual appearance. Thus, refinement using a list or a grid presentation, may have an even worse usability than immediate selection in dense environments.

Interaction techniques for manipulation in handheld AR often use the multi-touch capabilities of the device's touchscreen. To provide full 3D interaction by touch in an integral way, existing approaches use complex multi-finger gestures. However, they are difficult or impossible to use in handheld AR scenarios, as the input device is held by at least one of the user's hands. Thus, the number of fingers to interact with the screen is limited. Multi-finger gestures may suffer in ease-of-use due to the small touch screen, as well as the usability of the application in general, as a reasonable part of the screen is occluded by the user's fingers during interactions. Furthermore, gestures on a 2D input device either need to abstract interactions in 3D space or simplify them by reducing the simultaneous accessible dimensions. Strong abstractions require prior knowledge for their application whilst a simplification induces additional interaction steps to complete the intended interaction. Both do not provide an intuitive way to interact with the environment.

All interactions are intended to be used in a handheld AR environment in which it is necessary to hold the device with one hand and to operate the touchscreen with the other. Having only one hand at disposal further impedes all interactions with the AR environment and limits the complexity of input gestures that can be used for modeling, selection and manipulation of virtual objects.

1.3 Aim of the Work

The purpose of this work is to investigate and to design interaction techniques for *modeling*, *selection* and *manipulation* of virtual objects in handheld AR environments. The proposed techniques aim at making use of the features offered by modern handheld devices. Furthermore, the techniques aim to cope with the limitations a handheld device introduces to provide an improved user experience and higher efficiency. This work will address the following research questions that arise from the problems stated in the previous section.

- Which device gestures are suitable for intuitive modeling of captured 2D shapes for simple polygonal modeling?
- How can the problems of 3D selection in dense handheld AR environments be solved with the help of the handheld's touch input.
- How can the original spatial arrangement of multiple selected objects be preserved to avoid false selection?
- How can the handheld be used to perform intuitive 6DOF manipulations of selected content and which limitations does it face.
- How well are 6DOF manipulations, that intend to allow more intuitive interaction, understood and accepted compared to traditional handheld touch gestures.

1.4 Individual Publications

Results of this work have been published previously by the author. The following peer reviewed publications describe the preliminary outcome of the work:

- Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. DrillSample: precise selection in dense handheld augmented reality environments. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*, pages 10–20. ACM, 2013.
- Annette Mossel, Benjamin Venditti, and Hannes Kaufmann. 3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality. In *Proceedings of the Virtual Reality International Conference: Laval Virtual*, pages 12–22. ACM, 2013.

1.5 Outline

This thesis is organized as follows. Chapter 2 lists and discusses relevant related work of interaction techniques for modeling, selection and manipulation of objects and their application in one-handed handheld augmented reality environments. In Chapter 3, the author's methodology to overcome the stated problems is described and the novel interaction techniques are presented that were designed accordingly. Chapter 4 describes applications that were developed as part of this thesis emphasizing important implementation details. Chapter 5 describes the design and setup of a user study that was conducted to evaluate the proposed selection and manipulation techniques and presents the results. Furthermore, quantitative and qualitative performance measures for the novel modeling interaction are presented. The thesis closes in Chapter 6 with a summarisation of the author's work and findings. Furthermore, shortcomings are addressed that are subject of future work and research.

Related Work

This chapter covers the related work relevant for this thesis. Applications and techniques for polygonal modeling, especially focusing on current developments for handheld applications, but also techniques for desktop use, are presented. Manipulation is followed by state-of-the-art techniques for the three canonical manipulation tasks *Selection*, *Positioning* and *Rotation* according to Bowman [4] as well as object shape manipulation by *Scaling*.

2.1 Augmented Reality

Before the related work for the specific interaction techniques is addressed, this section will provide an introduction to *Handheld Augmented Reality* on which this thesis is constituted.

2.1.1 Definition & Characteristics

In Augmented Reality (AR) users are able to see the real world superimposed with virtual objects that behave as if they would actually reside in the real world. Azuma [1] defines the essential components of Augmented Reality with the following three characteristics:

1. Combines real and virtual.
2. Interactive in real time.
3. Registered in 3D.

The characteristics help to delimit AR from Virtual Reality (VR) or other technologies that look similar, such as 2D overlays. A video stream or a see-through display that is combined with 2D information, such as live temperature sensor readings, does not qualify as AR because the 2D information is not registered in 3D. In VR, a user is (visually) fully immersed in the Virtual Environment (VE) and is unable to see the real environment. In VR the real and virtual are not combined, while in AR a user is always able to perceive the real environment as well as the VE [1].

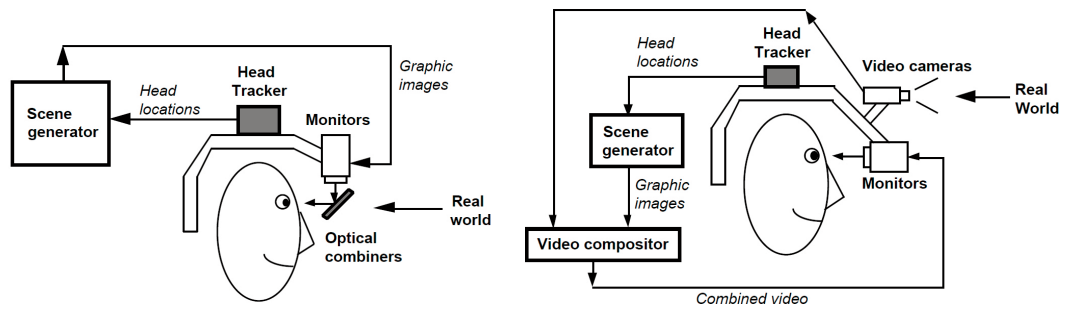


Figure 2.1: Conceptual diagrams of an optical- (left) and a video- (right) see-through HMD [1].

The augmentation in AR systems is either accomplished as an optical- or a video-see through setup. The conceptual difference is illustrated for a head mounted display (HMD). The key components of an optical-see-through HMD, as depicted left in Figure 2.1, are:

1. a head-tracker (input device),
2. a scene generator
3. and a monitor coupled with an optical combiner (output device).

The head-tracker is used to register the user's head-position and -orientation in 3D space. The scene generator uses the 3D position and orientation from the head-tracker to generate a photorealistic representation of the VE that is aligned with the real environment. Finally the monitor and the optical combiner is used to present and combine the real environment with virtual content. A video-see-through setup will use a video camera and a video compositor instead of an optical combiner [1].

2.1.2 Augmented Reality Interaction

If no further hardware is employed in an AR setup other than depicted in Figure 2.1, users can only interact with the VE by means of travel. Here, the tracking 3D pose of the head-tracker is used to travel within the VE. Travel means that users are able to move and change their orientation within the virtual environment. However, as the head-tracker (input device) and the augmented display (output device) are both fixed to the HMD, they form a single entity. Thus, only a single 6DOF pose is used for input and output at the same time.

To allow users to intuitively select and perform RST (Rotate-Scale-Translate [40]) manipulations on objects in the VE, another input device, such as a 3D pointer, is used in AR for interaction [53]. Figure 2.2 shows a wireless pen developed by Kaufmann [22] that is tracked using retro-reflective markers and an infrared-based optical tracking system. Using another tracked input device that is controlled by the users hand, allows the design of intuitive interactions as a second 6DOF pose is available.



Figure 2.2: A trackable wireless, one-button pen with retro-reflective markers [22].

2.1.3 Handheld Augmented Reality

Handheld (mobile) AR is similar to traditional AR with a video-see-through HMD. Wagner [53] defines handheld AR according to Figure 2.3(b-d), whereat a user has to actively hold the device in his hand. Nowadays, PDAs and mobile phones have merged into smartphones, however Wagner’s classification still applies, as smartphones are available in various sizes and form factors. There are two major differences of a handheld AR setup to a traditional AR setup. First, users have only one hand available to interact with the device. And second, as only the handheld is registered with the real world, there is only a single 6DOF pose available for interaction.

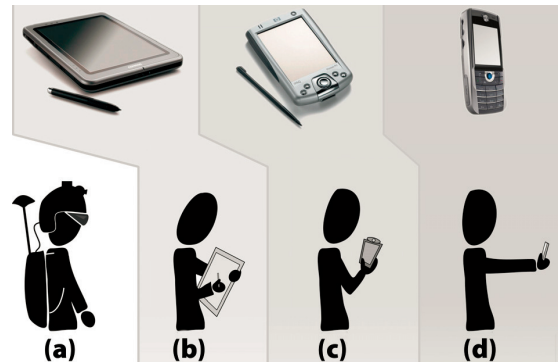


Figure 2.3: Form factors of Mobile Augmented Reality systems: (a) traditional “backpack” computer & HMD, (b) Tablet PC, (c) PDA, (d) Mobile phone [53].

In handheld AR, tracking can be accomplished by using various technologies, such as optical tracking, GPS, a compass, gyroscopes and accelerometers. Wagner concludes in [53] that using smartphones for handheld AR is highly intuitive and furthermore presents in [54] techniques for realtime natural feature tracking on mobile phones. Figure 2.4 shows a user interacting in a handheld AR setup that uses natural feature tracking. The position (x,y,z) and orientation (roll, pitch, yaw) (see Figure 2.5) of the handheld device is recognized in real-time and used to overlay the video stream of the back-facing camera with a perspectively correct representation of the virtual content, thus constituting the AR view.

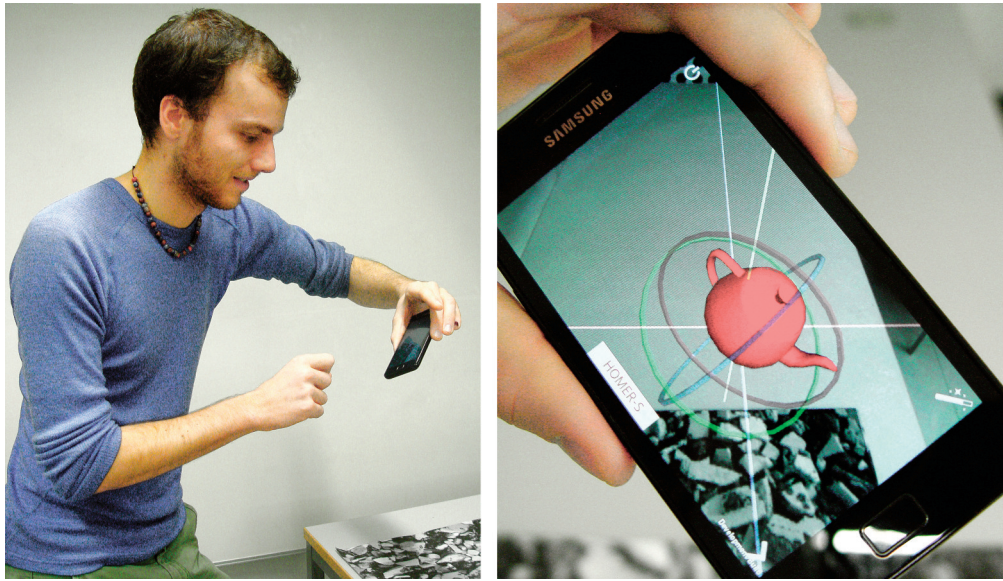


Figure 2.4: Interacting with a handheld AR setup (left). The AR view as seen by the user (right).

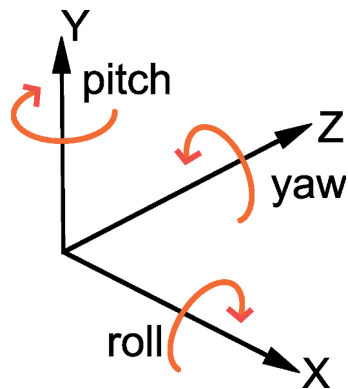


Figure 2.5: 3D position and orientation of a tracked input device.

2.2 Polygonal Modeling

Modeling is the act of creating a 3D surface of an imaginary or a real object. Such surfaces can be represented using a mesh that approximates the object's surface by subdividing it into triangles or quadrilaterals - the polygon mesh. There are various ways to create and manipulate meshes. The capabilities and the concrete interaction for modeling operations depends mostly on the employed modeling software. Therefore, this Section describes selected operations to create meshes without referencing concrete implementations.

2.2.1 Basic Modeling Operations

The creation of meshes using polygonal modeling often starts by using a primitive, such as a box, a sphere, a cylinder, a plane or a line. These primitives are then modified or combined with different operations to create a more complex surface. The lines and shapes can be expressed either as explicit sets of points or as parametric curves, e.g. NURBS [41]. Popular ways to create meshes are extrusion, sweeping and lathing [42].

Extrusion is a simple modeling operation available in all modeling software applications. It builds cylindrical meshes by adding depth to a flat 2D surface. The extrusion can be described easily by a displacement vector $\vec{d} \in \mathbb{R}^3$ that is added to each of the surface's vertices and allows the creation of oblique cylindrical meshes (see Figure 2.6a). More complex meshes can be created by defining a path (a list of displacement vectors) along which the extrusion takes place. The extrusion is applied for each point successively and creates a mesh consisting of cylindrical sections [42].

Sweep is a modeling technique that is very similar to extrusion along a path. However, the major difference is that extrusion creates segments with parallel cut surfaces whilst for sweeping, the surfaces can be rotated so that smooth curves can be modeled. See Figure 2.6b and 2.6a for illustration.

Lathe produces meshes that are radially symmetrical to a fixed axis. A lathed surface is defined by a flat shape or curve, the axis around the surface is rotated and the rotation that is applied. Figure 2.6c depicts a surface which is rotated around the pitch-axis by 180° .

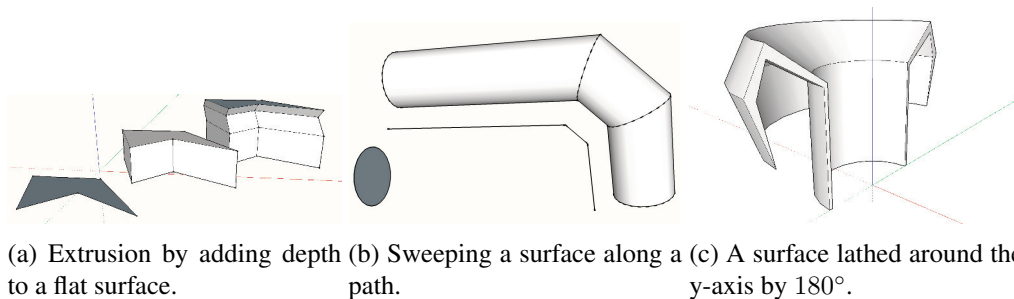


Figure 2.6: Extrusion, Sweep and Lathe.

2.2.2 Handheld Modeling Applications

The afore mentioned operations and manipulations are mostly used and designed for professional desktop modeling applications such as *AutoCAD*, *Blender* or *SolidWorks*. However, there are applications available for handheld devices as well. A few modeling applications that are currently available for handheld devices are *Sketcher 3D Lite/Pro*, *Spacedraw* and *TrueSculpt Virtual Sculpture*. The applications use different approaches for modeling on handheld devices.

Unfortunately, no applications that employ augmented reality as a concept for modeling were found.

Sketcher 3D Lite/Pro This application runs on Android devices and is rather a physics sandbox than a modeling application. Nevertheless, *Sketcher 3D* [47] provides an interface that is simple, intuitive and offers the basic functions to interact with virtual content such as:

- Create primitives (cube, sphere, tube, cone).
- Move objects by using simple touch gestures.
- Rotate and scale an object using the handheld’s accelerometer.
- Change an object’s color or lock it to disable unintentional editing.
- Create groups of objects and apply modifications to the all objects in the group.
- Duplicate already created and modified objects.

All functions regarding the manipulation of an object are accessible through a circular menu (see Figure 2.7) upon selection. New objects are created by double-tapping on the empty space in the scene which triggers a circular menu or for scaling uniformly as well.

Sketcher 3D only allows the creation of prefabs and cannot be regarded as a modeling application due to the lack of actual modeling functionality. However, its interface is a good example for handheld applications as it uses only simple touch input gestures to interact with the objects and therefore inspires modeling applications in one-handed handheld AR.

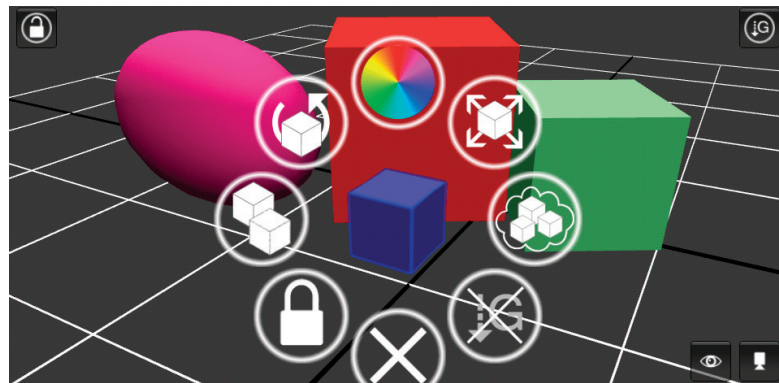


Figure 2.7: Circular action menu in *Sketcher 3D* [47].

Spacedraw The description in *Google Play*[44] describes Spacedraw as a “full-featured 3d-modeling software designed for tablets & smartphones“. It provides functionality that is available in professional modeling applications. It can create primitives (rectangle, circle, sphere, cube, line splines), dbut also model more complex shapes, such as prisms, pyramids, discs, cylinders, cones, tori, spheres or helices. Manipulations (e.g. extrude, delete, move) can be

applied per vertice or face separately. Aside from modeling operations, it supports texturing by applying solid colors or user-defined textures. The user interface is structured in a textual toolbar (see Figure 2.8). The menu is placed in the upper left corner and holds six ribbons (submenus). The first three ribbons let the user switch between model *creation*, model *manipulation* and model *texturisation*, whilst the remaining three ribbons are used to access the system settings *display*, *view* and *file*.

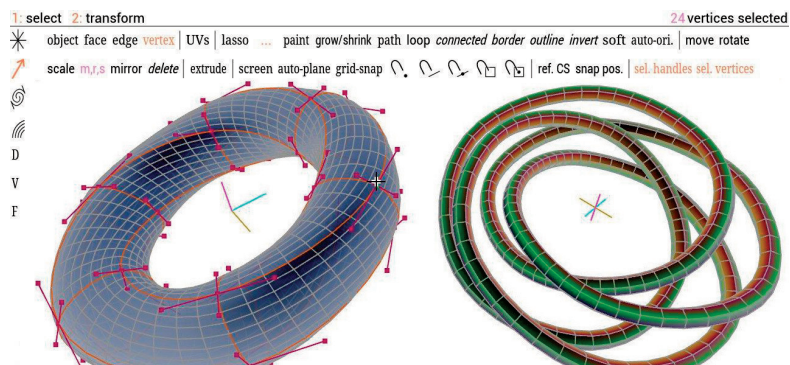


Figure 2.8: Textual command menu in *Spacedraw* [44].

The interactions used in *Spacedraw* to control the application are similar to the the toolbar widget used in graphic user interfaces for desktop applications. Using a toolbar on handheld devices is problematic. The screen space is limited and a permanently visible toolbar reduces it even more. Additionally, a toolbar is not intuitive, as it requires the user to be familiar with its terminology. *Spacedraw*'s toolbar is mostly textual but also contains a few icons and occupies approximately 5% of the screen. The buttons in the toolbar are comparably small, which makes it difficult to use. Furthermore, many actions are hard to understand, as it is not clear to which objects (vertice, edge, face, object) they can be applied to, how they are applied or what effects they have. It is possible to draw a freeform shape in 3D space that can be used in a second step as primitive for modeling. Drawing primitives suffers greatly in precision on the small screen due to the large area covered by a fingertip. As the shape is drawn in 3D space, it can be subject to unintentional dislocations caused by the 2D/3D mapping of the projection on the screen.

TrueSculpt Virtual Sculpture Unlike the previous two applications, *TrueSculpt Virtual Sculpture* [7] is designed for modeling using digital sculpting. It is suited best for modeling organical structures (see Figure 2.9) instead of geometrical surfaces. Digital sculpting intends to resemble the process of sculpting materials, such as clay. Modeling an object is performed by manipulating the surface of a predefined object (e.g. a sphere) with a given tool. Available tools are inflate/deflate, grab, smooth, flatten, pinch and noise and are all adjustable by its radius of impact and strength. The user interface is comparably simple as it only allows to switch between tools and to change a tool's parameters.

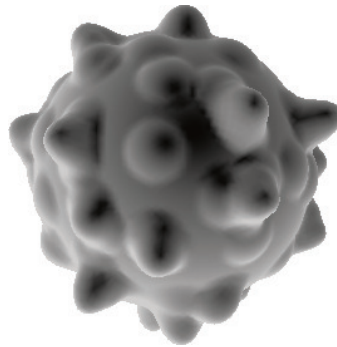


Figure 2.9: Bumps on a sphere created with *TrueSculpt Virtual Sculpture* [7].

2.3 Selection

The process of indicating and confirming a desired object is called selection. The indication should be supplemented by visual, aural or haptic feedback, when choosing a certain object from a set of others. One of the most natural ways to indicate an object in augmented reality environments are techniques that use the pointing metaphor, such as *Ray-casting*.

2.3.1 Ray-casting

One of the earliest implementations of the Ray-casting selection technique can be found in [2]. Here, the user indicates an object to interact with by simply pointing at it. A line along the user's arm performing a pointing gesture is casted into the virtual environment and the closest object intersecting this line is used for the desired interaction. In fully immersive environments, the line can be defined either a) from the user's head to a sensor attached to its hand or b) from the user's head into its gazing direction. While Ray-casting is a simple and powerful selection technique for objects that have a large visual appearance on the image plane, it fails to work for objects that are small in general or are far away and thus result in a small visual projection on the image plane. This problem is mostly introduced by the high angular accuracy that is necessary for selecting small objects. Even worse, a small angular change induced, e.g. by tracker or hand jitter, causes a strong spatial disgression at far distance. Figure 2.10 shows the Ray-Casting technique being used to point at a solitary sphere [4, 35].

2.3.2 Handheld Ray-casting adaptations

Telkenaroglu and Capin present in [51] a set of 3D interaction techniques for handheld devices with a touchscreen. Their major design objectives were to overcome the problems of target occlusion as well as the limited precision caused by the area a finger tip covers on the comparably small device. Therefore, they proposed two techniques based on the popular Ray-casting selection technique.

The *Dual-Finger Midpoint Ray-Casting* technique is operated with three fingers. From the midpoint C_{mid} between two fingers (see Figure 2.11), at the touch points f_1 and $f_2 \in \mathbb{R}^2$ on the touchscreen, a Ray-Cast is performed towards the virtual environment (as in Equation

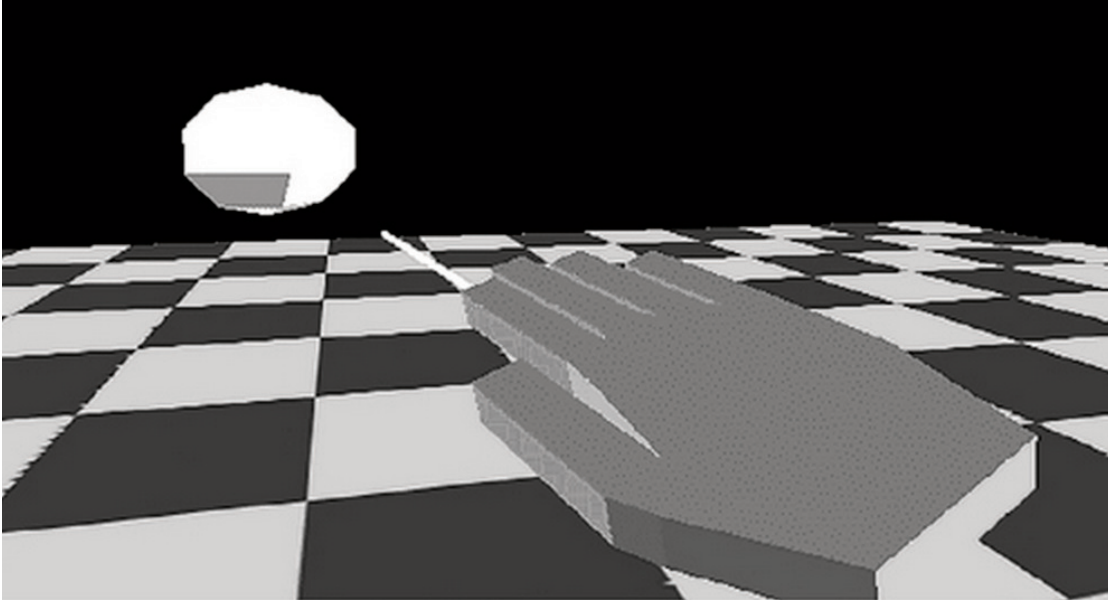


Figure 2.10: Ray-casting selection technique [35].

2.1). During the selection, a cross-hair is displayed at C_{mid} to assist the selection. The first intersecting object is then highlighted to indicate a possible selection. The selection is confirmed with an arbitrary third touch on the screen. Furthermore, to increase the precision for highly occluded objects, the view can be zoomed in and out at the *midpoint* by moving one of the two fingers away from the other or bringing them closer together respectively.

$$C_{mid} = \left(\frac{f_1.x + f_2.x}{2}, \frac{f_1.y + f_2.y}{2} \right) \quad (2.1)$$

$$C_{off} = (f_1.x + o.x, f_1.y + o.y) \quad (2.2)$$

The *Dual-Finger Offset Ray-Casting* technique is, unlike the previous one, operated with only two fingers. By default this technique uses a predefined offset $o \in \mathbb{R}^2$ to place the crosshair above the user's finger (see Figure 2.12) at C_{off} as in Equation 2.2. One finger is used to move the crosshair on the screen from which Ray-casting is performed. Whenever an object is hit by the Ray-cast, it is highlighted immediately. The second finger is used to either zoom the view, modify the offset o of the cross-hair or to confirm the selection. The three interactions are triggered depending on the distance d between the two touch points f_1 and $f_2 \in \mathbb{R}^2$ in relation to a given distance threshold t_d . If an object is highlighted, a second finger touches the screen and $d < t_d$, the highlighted object is selected. If only the second finger is moved on the screen and $d > t_d$, the offset o is modified to match the midpoint between the two touchpoints. If both fingers are moved and $d > t_d$, the view is zoomed to increase the precision.

Both techniques tackle and overcome the problems of partly occluded targets and reduced precision that arise when using Ray-casting on handheld devices. However, these techniques are deemed to be suboptimal for one-handed handheld AR scenarios for various reasons. Near the screen's corners and borders, the *Dual-Finger Midpoint Ray-Casting* fails to operate as the

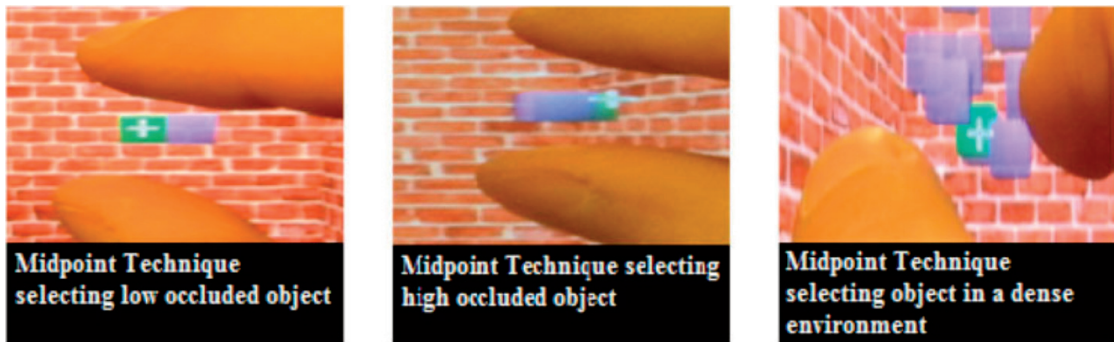


Figure 2.11: *Dual-Finger Midpoint Ray-Casting* selection technique [51].



Figure 2.12: *Dual-Finger Offset Ray-Casting* selection technique [51].

fingers would need to be detected outside of the touchscreen. Generally, by using multiple fingers, large portions of the touchscreen are occluded and important information of the targeted object's surrounding environment is not visible. The already limited interaction space on the handheld's touchscreen is further reduced, as up to three fingers have to fit on the screen. Finally, the gestures are harder to apply due to their comparably high complexity.

2.3.3 SQUAD

The *SQUAD* selection technique was introduced by Kopper, Bacim and Bowman [23] as a rapid and accurate 3D selection method that employs progressive refinement. *SQUAD* is an acronym for **S**phere-casting refined by **Q**UAD-menu. The main idea is to overcome the problem of selecting small objects with Ray-casting by using a Sphere-cast [23] instead and multiple steps for refinement. Thus, the selection process is split into two phases. In the first phase, Sphere-casting is used to define the set of objects containing the object of interest. Sphere-casting uses a Ray-cast to determine, as the closest intersecting object of the Ray-cast determines at which position a sphere is casted. The size of the sphere is adjusted according to the distance of the closest intersection. All objects intersecting the sphere are subject of the refinement in the subsequent step. In the second phase, the image plane is divided into four equally sized areas, the quad-menu, in which all objects of the Sphere-cast are evenly distributed neglecting their original spatial position. Users can now progressively narrow down the available objects by choosing

a quadrant from the menu repeatedly. Each time a quadrant is selected, the objects placed in that quadrant are rearranged evenly on all four quadrants. Therefore at least $\lceil \log_4(n) \rceil$ selection steps are necessary to select an arbitrary object from n objects of the initial Sphere-cast in the first phase. An illustration of the two steps is found in Figure 2.13.

By employing a volumetric cast, *SQUAD* overcomes the difficulty of precise selection of small objects at the cost of multiple interaction steps. However, [23] states that *SQUAD*'s is only more efficient than Ray-casting if the environment's density is not too high and the size of the objects is not too small. Additionally, *SQUAD* removes the selected objects from their original spatial context during refinement. Thus, false selections will occur if the desired object is not visually distinguishable from its surrounding objects.

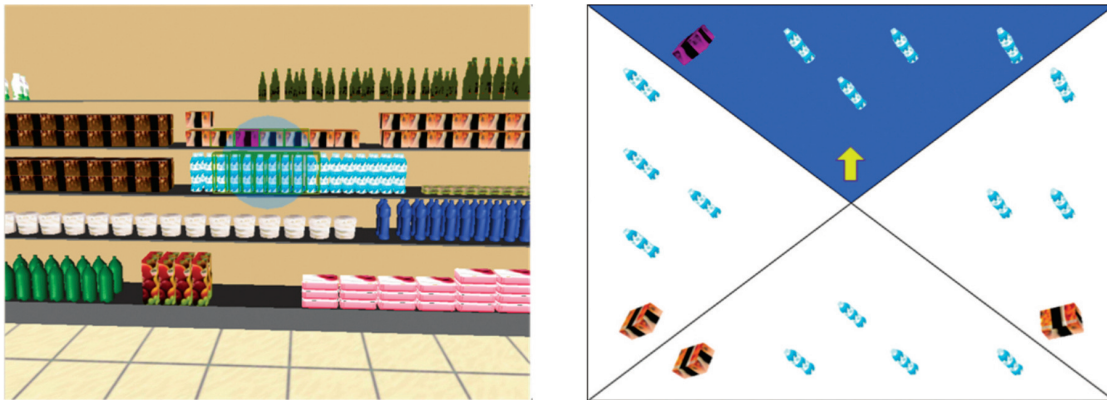


Figure 2.13: *SQUAD* selection technique: The left image shows the impact of the Sphere-cast in the first phase. The right image shows the arrangement of the objects in the QUAD-menu [23].

2.3.4 EXPAND

Introduced by J. Cashion, C. Wingrave and J. LaViola in 2012 [10], the design of *Expand* was driven by the problems that *SQUAD* induces when it removes the objects from their original context during refinement. Their idea is to use a dynamically sized grid that fills the screen instead of a fixed QUAD-menu. This allows for a spatial relocation of the selected object resembling their original spatial arrangement and therefore providing more clues to identify the desired object. Furthermore, the progressive refinement is omitted and an animation, to support the understanding of the spatial context relocation, is introduced. The first phase of the selection process is similar to *SQUAD* but a *Cone-cast* [25] is used to define the selection volume instead of a Sphere-cast. At the beginning of the second phase, all objects intersecting the cone volume are cloned. In a transition animation, the clones are moved from its original position to their designated position on the virtual grid filling the screen. The clones are arranged on the grid to reflect the spatial context of its original counterparts. In the third phase, after the transition has ended, the user can immediately select without any further refinement one of the objects by pointing at them by employing a simple Ray-cast. Figure 2.14 shows *EXPAND*'s grid

alignment of multiple selected objects as well as in the background the environment they were selected from.

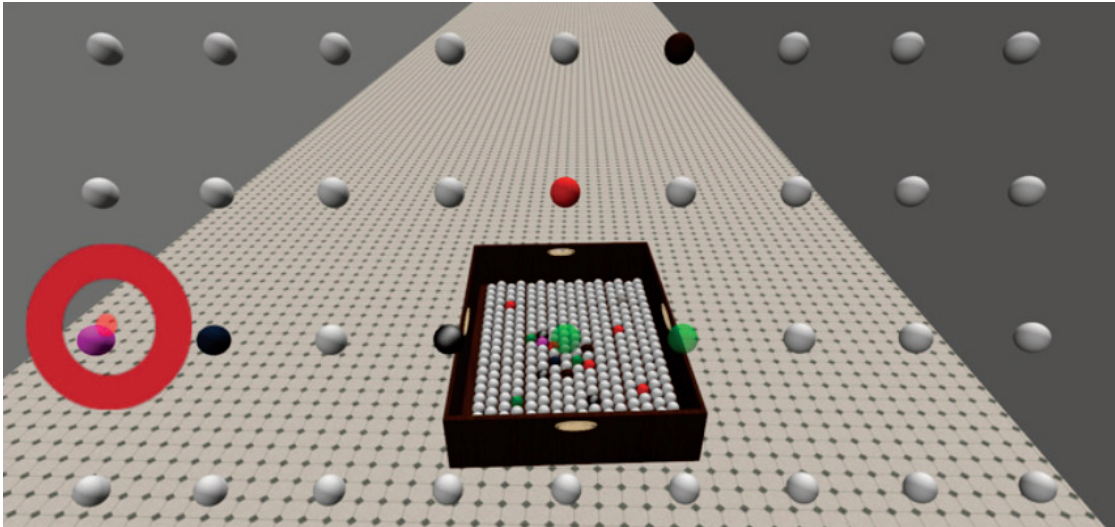


Figure 2.14: *Expand* selection technique: The original objects are seen in the background. The foreground shows the rearranged clones on the virtual grid [10].

Expand could eliminate the, possibly cumbersome, progressive refinement by displaying all selected objects at once on a dynamically sized grid. For a selection from a set of well arranged and clearly visible objects, as seen in Figure 2.14, *Expand* is capable to work well. Unfortunately, it is not further specified in [10] how the objects are mapped with $f: R^3 \rightarrow R^2$ on the grid. It is uncertain if the mapping f resembles the original 3D arrangement close enough to avoid false selections for objects with a similar or identical visual appearance, especially when the objects are partially or fully occluded.

2.4 Manipulation

There are many manipulation techniques that have been developed for augmented reality applications ranging from exocentric metaphors like the World-in-miniature [48] technique to egocentric metaphors like *virtual hand* [4]. In exocentric techniques, a user interacts with the virtual environment from outside whilst with egocentric techniques a user interacts with the VE as part of it from a “first-person view”. A short overview of egocentric metaphors is given in Figure 2.15. As only few techniques are applicable in one-handed handheld AR setups, this thesis focuses on egocentric techniques and contains a detailed description of the virtual hand technique and state-of-the-art techniques for handheld environments.

2.4.1 Virtual Hand

3D manipulation techniques that map the motion of the user’s hand to the motion of a virtual hand, and thereby constitute a direct relationship between virtual-world and real-world coordi-

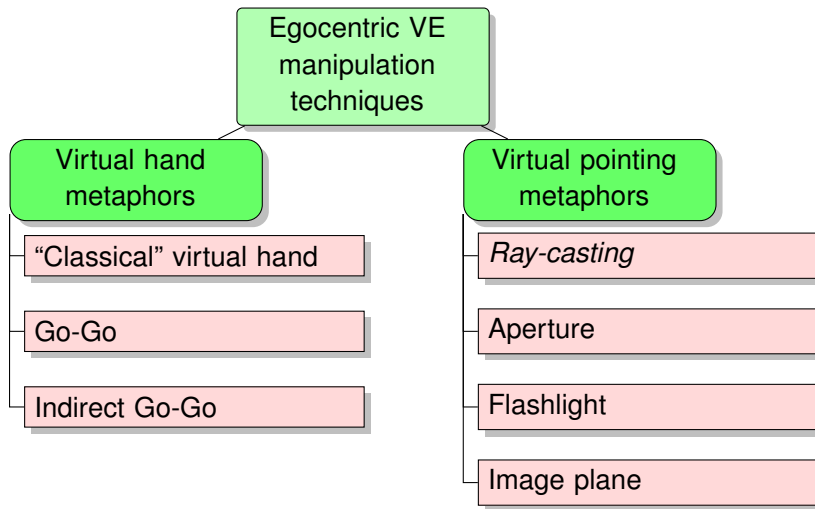


Figure 2.15: Manipulation techniques classified by metaphor according to Bowman [4].

ates, are summarised as *virtual hand* manipulation techniques. Often, virtual hand techniques use a 3D pointer to visualise the position and orientation of the virtual hand in the VE (see Figure 2.16a) [4].

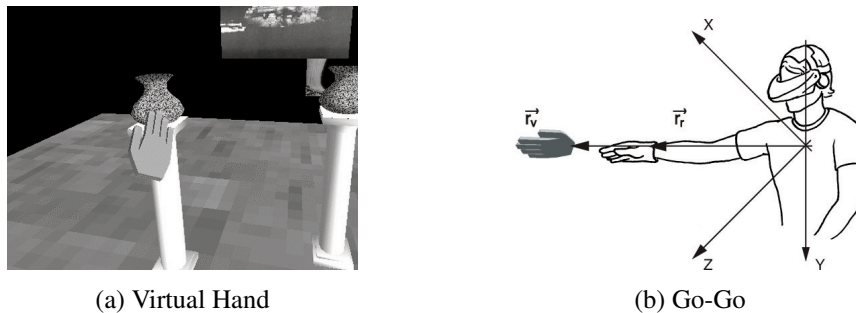


Figure 2.16: Manipulation techniques using the virtual hand metaphor [36].

The simple virtual hand manipulation technique directly maps the user's hand movement and rotation to the virtual hand. The relationship between a state of the real hand S_r and the virtual hand S_v are described as a transfer function in Equation 2.3. The position of the virtual hand p_v is directly derived from the user's hand position p_r multiplied with factor α (first-order transfer function). The rotation of the user's hand R_r is directly applied to the virtual hand's rotation R_v (zero-order transfer function). The mapping with α may be necessary to align the working volume of the VE with the working volume of the real world and causes in Equation 2.3 the virtual hand to move faster than the user's hand. Virtual hand manipulation techniques are classified as isomorphic interaction techniques due to the linear mapping they use. This allows for very intuitive manipulations with the virtual environment as the interaction of real world objects is mimicked. However, the linear mapping has a severe drawback caused by the limited

length of the user's arm whenever objects out of its reach need to be manipulated [4].

$$p_v = \alpha p_r \text{ and } R_v = R_r \quad (2.3)$$

Figure 2.16b shows the virtual hand used in the *Go-Go* interaction technique [36]. *Go-Go* is similar to simple virtual hand but improves it by extending the reach of the virtual arm with the help of a non-linear mapping function.

The *HOMER* (Hand-centered Object Manipulation Extending Ray-casting) interaction technique by Bowman and Hodges [5] was developed to overcome the problems of virtual hand techniques. The name already reveals that it uses Ray-casting for object selection and uses simple virtual hand for object manipulation. Therefore, it can be classified as a combined egocentric selection & manipulation technique. Upon selection the virtual hand travels to the selected object and is attached to it. Subsequently, the user performs manipulations and upon deselection, the virtual hand travels back to the position where it was before the selection. With the help of the selection by Ray-casting, the virtual working-volume no longer needs to be adjusted with a fixed α , as it was done with the simple virtual hand. With *HOMER*, the factor α is defined when an object is selected as the ratio of the real hand's distance and the virtual object's distance, thus constituting a dynamic mapping during the manipulation. The function describing the virtual object's translation are given in Equations 2.4 and 2.5. The factor α_h is calculated upon object selection with D_o as the distance of the virtual hand to the selected object in the virtual world and D_h as the distance between the user and its hand. The transfer function to manipulate the rotation remains the same as in Equation 2.3.

$$p_v = \alpha_h p_r \quad (2.4)$$

$$\alpha_h = \frac{D_o}{D_h} \quad (2.5)$$

Generally, virtual hand techniques enable intuitive 6DOF manipulation (translation and rotation) and were in [5] found to be easy to use and natural due to the mapping of the real hand to the virtual hand. With *HOMER*, users can easily reach even distant objects and translate them in the virtual world. However, the virtual working volume depends on the relation in Equation 2.5 and might therefore be limited. To put an object very far away from its original position, multiple manipulation attempts might be necessary.

2.4.2 Z-Technique

The *Z-Technique* is a 3D positioning technique for multi-touch displays proposed by Martinet et al. [28]. This technique is designed to use separate hands for the manipulation of objects. The interaction is started with the user's dominant hand pointing at an object visible on the touchscreen. Hence, *Z-Technique* uses Ray-casting for selection. Upon selection, the same finger is used to translate the object on a 2D-plane coplanar to the image plane positioned at the center of the selected object's. The depth position of the object can be controlled with a second finger touching the screen. Given the user's position, a relative motion is calculated

from the movements of the second finger on the screen. Moving the finger away from the user translates the object away from the user's view and vice versa. Furthermore, a non-linear mapping function is used to translate the object back and forth. For a strong finger displacement of successive movements, a higher translational scaling factor is used and a smaller one for low displacement, thus enabling the user to position an object precisely but also at a large scale. The interaction ends as soon the first finger, which was used for selection, is released from the surface. Figure 2.17 illustrates its use. The Z-Technique was evaluated by Telkenaroglu in [51] for handheld devices and was found to be "easy to position objects on screen locations ..." and "precisely to position objects ...". However, it does not provide rotation of objects and requires both hands for its operation, so that its application in one-handed handheld AR scenarios seems limited.

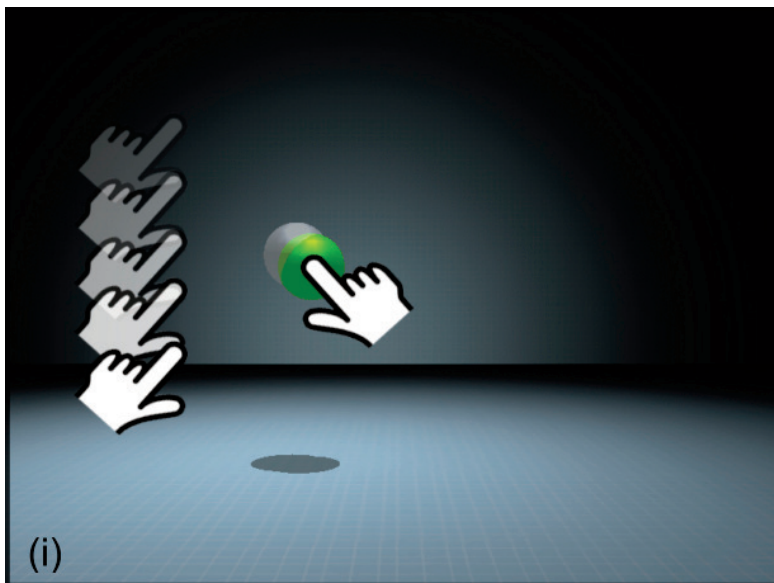


Figure 2.17: „Illustration of the Z-Technique. The first finger (right hand in the example) is used for direct positioning in the camera plane while the second finger (left hand) is used for depth positioning in an indirect way. Backward-forward movements move the object farther or closer to the user [28].“

2.4.3 Dual-Finger mobile 3D interaction techniques

Telkenaroglu and Capin introduced in [51] adaptations for selection by Ray-casting but also presented interaction techniques to manipulate virtual objects with two fingers. They present the techniques *Dual-Finger Midpoint Translation*, *Dual-Finger Rotation* and *Dual-Finger Scaling* that decouple the corresponding 3DOF manipulation into smaller subtasks. Figure 2.18 illustrates the user interface for moving and rotating an object. *Translation* is split up into two modes of translation. First, along the y-axis and second, on the xz-plane of the world coordinate system. If two fingers are adjoined, i.e. the distance d to each other is less than 100 pixels, and they

are moved up and down on the input screen, the selected object is translated on the y-axis. If the two fingers are apart, i.e. $d > 100$ pixels, a cross-hair is displayed between the fingers and the object can be translated on the xz-plane. The object's new position P is calculated by the intersection E on the xz-plane with a ray casted towards the cross-hair's position on the screen, with $P = (E.x, P.y, E.z)$ retaining its original y-position. *Rotations* are performed with two fingers and for each axis separately. Moving the fingers horizontally on the screen, rotates the object around the pitch-axis and moving them vertically, rotates the object around the yaw-axis. Moving the fingers in opposite directions (horizontal or vertical) results in a rotation around the roll-axis. Likewise, an object is *scaled* on each axis separately. For scaling along the x-axis, a horizontal pinch gesture is used and for scaling along the y-axis a vertical respectively. Scaling along the z-axis is performed by moving two fingers, positioned next to each other, vertically on the screen.

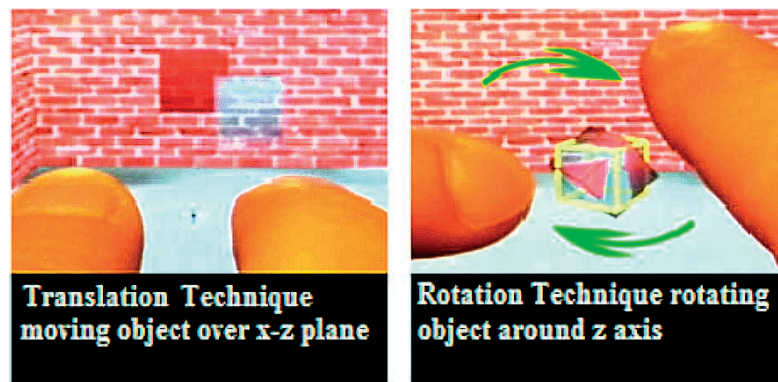


Figure 2.18: The userinterface while interacting with *Dual-Finger Midpoint Translation* and *Dual-Finger Rotation* [51].

The *Dual-Finger* techniques allow for precise manipulation of translation, rotation and scaling of objects by decoupling 3DOF tasks into 1-2DOF subtasks and assigning specific touch gestures. However, the use of multi-touch gestures is flawed (limited display size, more difficult to use) on handheld devices, as mentioned before in Section 2.3.2. Furthermore, the introduced gestures for the 1-2DOF subtasks are assigned inconsistently. For instance, the gesture “*move two fingers, positioned next to each other, vertically on the screen*”, triggers a translation along the y-axis but a scaling along the z-axis. That means that depending on the higher manipulation task (translate, rotate, scale), the same gesture has an impact on different dimensions of the object.

2.5 Touch Gestures

Gestures allow users to interact with applications by manipulating screen objects. The following table shows the core gesture set that is supported in Android. The Figures 2.19-2.26 as well as its description was taken from the Android developers guide [18].

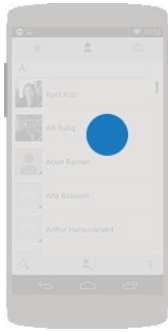


Figure 2.19: Touch.

Triggers the default functionality for a given item.

Action: Press, lift.

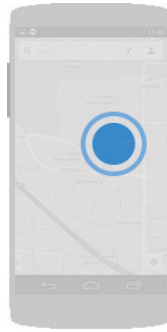


Figure 2.20: Double touch.

Scales up a standard amount around the target with each repeated gesture until reaching maximum scale. For nested views, scales up the smallest targetable view, or returns it to its original scale. Also used as a secondary gesture for text selection.

Action: Two touches in quick succession.

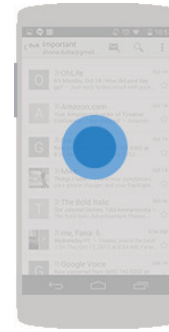


Figure 2.21: Long press.

Enters data selection mode. Allows you to select one or more items in a view and act upon the data using a contextual action bar. Avoid using long press for showing contextual menus.

Action: Press, wait, lift.



Figure 2.22: Pinch open.

Zooms into content.

Action: 2-finger press, move outwards, lift.



Figure 2.23: Pinch close.

Zooms out of content.

Action: 2-finger press, move inwards, lift.



Figure 2.24: Long press drag.

Rearranges data within a view, or moves data into a container (e.g. folders on Home Screen).

Action: Long press, move, lift.

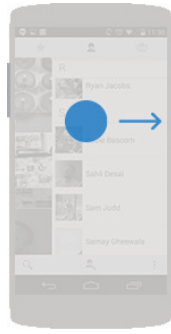


Figure 2.25: Swipe or drag.

Scrolls overflowing content, or navigates between views in the same hierarchy. Swipes are quick and affect the screen even after the finger is picked up. Drags are slower and more precise, and the screen stops responding when the finger is picked up.

Action: Press, move, lift.



Figure 2.26: Double touch drag.

Scales content by pushing away or pulling closer, centered around gesture.

Action: A single touch followed in quick succession by a drag up or down:

- Dragging up decreases content scale.
- Dragging down increases content scale.
- Reversing drag direction reverses scaling.

Methodology

This chapter outlines the investigated concepts and developed algorithms for modeling, selection and manipulation interactions in one-handed handheld AR scenarios to overcome the limitation of state-of-the-art techniques. Requirements are derived to form guidelines that support the design of appropriate interaction techniques. As results, the novel interaction techniques *Prefab Based Modeling* for modeling, *DrillSample* for selection and *3DTouch* and *HOMER-S* for manipulation are introduced.

3.1 Requirements

One-handed handheld AR introduces certain requirements to the design of modeling, selection and manipulation techniques. The careful consideration of requirements is important as they influence the efficiency and the ease-of-use of the techniques. These requirements are listed as follows.

1. **Single I/O device:** In handheld AR input and output device form a single entity, whereas immersive VEs provide separate devices to determine interactions (e.g. by tracking a wand) and the user's position and orientation (e.g. with a headtracker). That means, for handheld AR is only single 6DOF pose available and it is used for input and output at the same time.
2. **Limited gesture complexity:** When interacting with the handheld's touchscreen, the user's finger and a part of its hand will occlude a portion of the screen and thus block information on the screen. This flaw gains significance as the touchscreen is already of comparably small size. Furthermore, only one hand can be used for touch interactions as the other is required to sustain the user's view onto the VE by holding the handheld device. Both facts disallow the use of complex multi-finger and -hand gestures.

3.2 General Guidelines

From the requirements stated in the previous Section as well as from the motivation and problems stated in Chapter 1, two general guiding principles are derived. The principles are applicable for modeling, selection and manipulation techniques.

1. Keep direct touch abilities: The direct touch behaviour of common touch-interfaces should be maintained, i.e. interacting with an object or user interface control by touching its 2D representation on the touchscreen with the finger. Offset or midpoint interactions, as described in Section 2.3.2 and 2.4.3, are considered to be insufficient due to their limitations but also because they break with already established behaviour.
2. Keep interaction simple: Interactions with the touchscreen should be as simple as possible. Multi-finger interactions are generally neither intuitive metaphors nor necessarily well known. Therefore, one-finger input should be applied where possible. If multi-touch input is deemed beneficial, it should comply with already established gestures such as stated in Section 2.5.

In the following three sections, specific principles are derived additional for each interaction and the accordingly designed techniques are presented.

3.3 Modeling

3.3.1 Guidelines

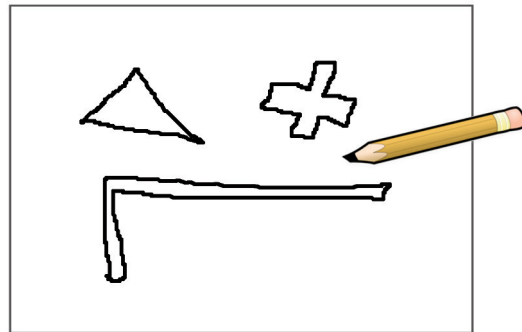
For the design of the modeling techniques further guidelines apply:

1. Keep interaction simple: *as stated in Section 3.2.*
2. Keep drawing intuitive: Primitives, such as lines and shapes, are commonly used for polygonal modeling and should be as easy as possible to create. Drawing in 3D space using a handheld device is difficult due to the limited screen size and the 2D touch input resulting in a 3D shape. Drawing primitives should therefore remain a 2D interaction and if possible be applied with high precision, for instance by using a stylus.
3. Keep modeling expressive: Modeling interactions on handheld devices should be as simple to understand as possible. Choosing a modeling technique from a textual interface that has a specific terminology is not intuitive. Instead the interface should express the capabilities of a modeling technique implicitly.

3.3.2 Prefab Based Modeling

Generally, modeling starts with drawing a 2D primitive which is then manipulated in a second step to create a 3D surface. *Prefab Based Modeling* maintains this process but aims at simplifying it. The idea is to provide a simple tool that allows the effortless creation of primitives for later 3D modeling. Instead of drawing primitives using the handheld itself, the primitives can

1a) Draw shapes with pen & paper.



1b) Capture shapes with handheld device.



2) Model 3D surface by extrusion.

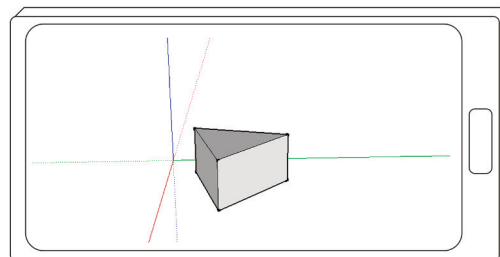


Figure 3.1: *DrillSample's* two-step selection process.

be drawn with pen and paper. The drawn 2D shapes are then captured with the camera of the handheld and saved as *prefabs* for later use to model 3D shapes. The actual modeling takes place in an AR environment and supports the idea of prefabrication as well. Here, users can choose from a set of predefined modeling techniques and combine these with previously captured 2D shapes. The illustrations in Figure 3.1 depict the different steps of the modeling process.

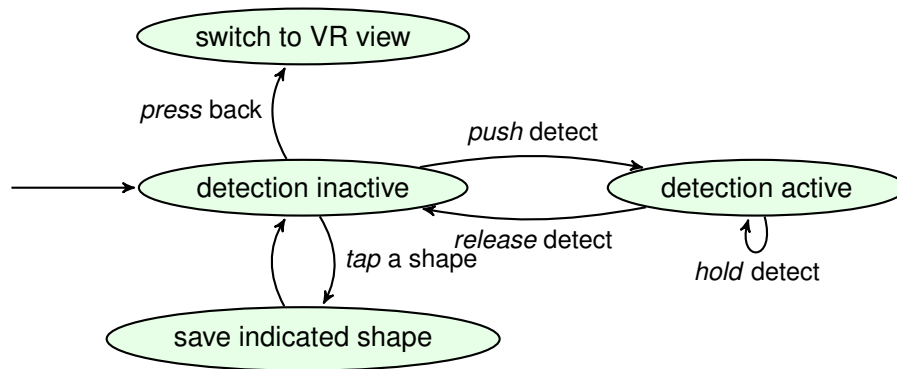


Figure 3.2: State diagram for the shape detection interface.

3.3.2.1 Capturing Shapes

In the first phase, shapes are drawn by hand on paper. Then the handheld is used to capture them using the built-in camera and image processing algorithms. Capturing the shapes is similar to recording a video with the handheld device. The user has to push and hold a button on the touchscreen while pointing with the back-facing camera to the surface on which the shapes were drawn. The video stream captured by the camera is displayed on the touchscreen without delay and acts as an AR view. The detected shapes are visually augmented in the AR view by visually emphasizing them to indicate their detection. Once the desired shape is detected, the user releases the button to stop the capturing process. The AR view is paused, the last video frame is displayed as a still image and is superimposed with the detected shapes. With a single tap gesture the shapes can now be saved to be used for modeling in the following phase. Drawing the shapes is intuitive by nature whilst capturing the shapes is designed to be as simple as possible. The *view-to-find* interaction used for capturing the shapes is simple and intuitive as users should be familiar with the interaction from recording a video. Additionally, the view-to-find interaction is used to avoid the video feed from being blurry, caused by the impact of a separate touch on the screen, when stopping the shape detection. Lifting resting finger, instead of tapping on the screen, introduces no shock to the handheld. The state diagram in Figure 3.2 illustrates the user's input and the resulting actions.

3.3.2.2 Perspective Driven Modeling

In the second phase, a user switches back to the AR environment to start modeling. Here a shape is chosen and then modeled with either *extrusion* or *lathe*. Whenever the shape or the modeling technique is changed, an exemplary modeled 3D surface is displayed in the AR view that represents the current selection. The model helps to understand the modeling technique's capabilities by its appearance so that the controllable parameters can be derived. Modeling is then performed by controlling the parameters of the modeling technique with simple touch gestures. Each of the technique's parameters can only be controlled separately and the exemplary model is modified in real-time to provide a vivid feedback. Displaying an exemplary model in the AR view supports the guideline to *keep modeling expressive* as users can inspect the resulting

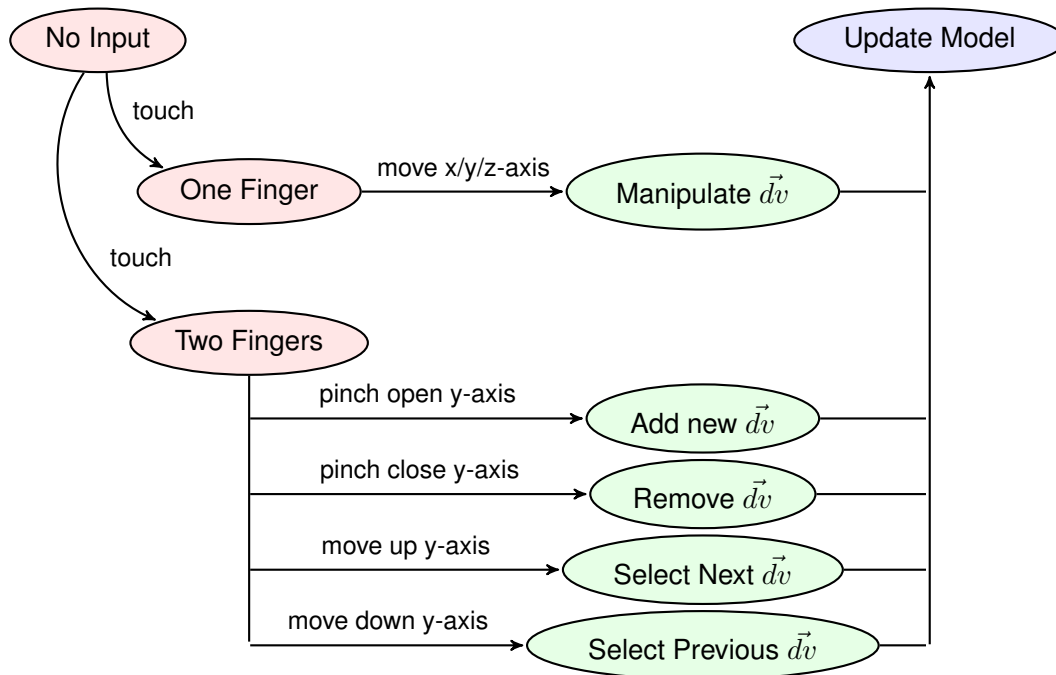


Figure 3.3: Interaction diagram for one-handed handheld modeling by extrusion.

model of a 2D surface they have drawn combined with a given modeling technique. Inspecting a detailed 3D model is much more self-explanatory than a textual or iconic menu.

The 3D models that are created with *extrusion* or *lathe* can, unlike freeform surfaces, be described with a few parameters. In common modeling techniques, the parameters of a model are not accessible intuitively as a user has to specify which parameter to modify or which tool to use to explicitly manipulate the model. When adjusting a parameter, it is important that the modification can be observed well, so that the adjustment forms a controlled interaction. Therefore, the parameters that are reasonable to adjust highly depends on the current perspective a model is examined in. For adjusting, e.g. a cube's height, it is preferred to choose a perspective that shows it from its side rather than from a bird-eye perspective. To make the parameters accessible in an intuitive way they, are mapped to the dimension of the modeled surface that follows the expansion. Accessible parameters are determined by considering the handheld's position and orientation by calculating the collinearity of the 2D input on the touch screen with the unit vectors of the model's coordinate system in world space. Section 3.5.3.1 describes the steps to calculate the collinearity in detail. Perspective driven modeling only allows the manipulation of one parameter at a time to simplify the interaction. It uses no abstract input like menus, buttons or textual input to maximise the screen space for the AR view. All actions are controlled by one or two finger touch gestures and are accessed by inspecting a pre-modeld surface. The mapping of the 2D input on the touchscreen to the parameters of the modeling techniques *extrusion* and *lathe* is described in the following paragraphs.

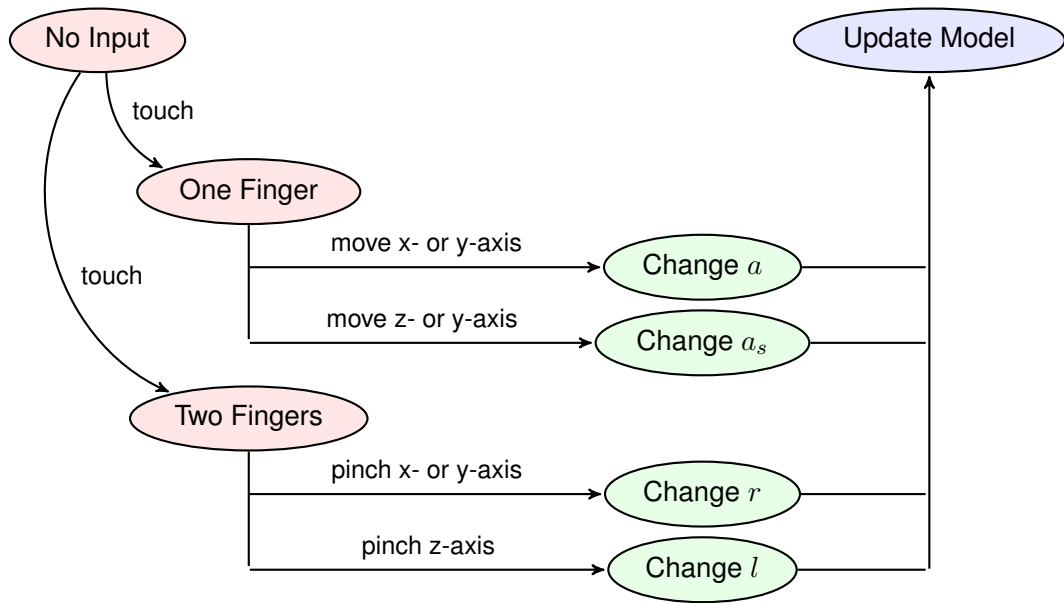


Figure 3.4: Interaction diagram for one-handed handheld modeling by lathing.

Extrusion is a basic modeling technique to create a 3D surface by adding depth to a flat surface. This was shown in the example in Section 2.2, Figure 2.6a. An extrusion can be fully described by a 2D shape and a set of displacement vectors that are used to build the connected cylindrical sections. Therefore, basic interactions to build a surface using an extrusion are: *a*) manipulate the coordinate of a given displacement vector $\vec{d}v \in \mathbb{R}^3$, *b*) select a specific $\vec{d}v$ and *c*) add or remove a $\vec{d}v$. All three actions can be mapped intuitively to simple input gestures. The position of a selected displacement vector is controlled with a single finger by moving it along one of the main axes. The x, y and z component's of $\vec{d}v$ are therefore controlled separately. All other actions are performed with two fingers. Moving two fingers into the same direction (scroll gesture) along the y-axis selects the next/previous $\vec{d}v$. A pinch open gesture (see Section 2.5) along the y-axis adds a new displacement vector at the current position of the extrusion and a pinch close (see Section 2.5) gesture removes the current $\vec{d}v$ respectively. Figure 3.3 shows the application of the input gestures along the axes to control the extrusion's parameters.

Lathe The second modeling technique is designed to create models by latheing. A lathe can create axis-symmetrical surfaces like tori. An example is given in Figure 2.6c. The lathe used in this technique is described by a 2D shape, the distance r (radius) to the axis it is revolved around, the amount of rotation α around the axis, the amount of the shape's rotation β around itself and a length d to stretch the lath into a screw-like shape. All parameters can be mapped using only one and two-finger gestures (see Figure 3.4). The two-finger pinch gesture is used to control the radius r (x- or y-axis) and the length d (z-axis) of the lathe. The rotations α and β are manipulated with one finger. However, they cannot be mapped to a fixed axis as they do not cause the model to expand along a single direction. Instead, the parameters α and β are,

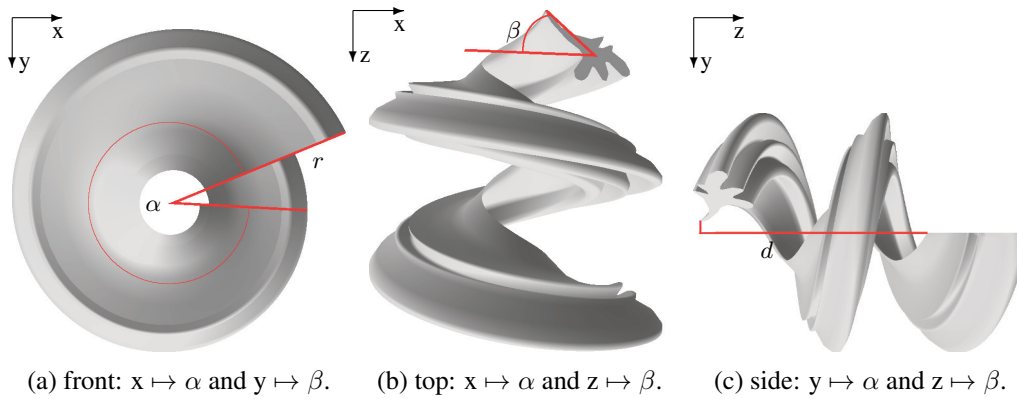


Figure 3.5: Perspective dependent mapping of one finger input for creating a lathe.

depending on the current perspective, mapped to different axes, as shown in Figure 3.5. When viewed from the front (x/y plane), touch input along the x-axis changes α and along y-axis β . From above (x/z plane), input along the x-axis controls α again, but as the y-axis is no longer accessible input along the z-axis controls β . Viewed from the side (x/y plane), α is mapped to the y-axis and β to the z-axis. To distinguish the perspectives the dot product of the handheld's look-direction \vec{d}_H in the world space with each of the world's unity vectors $e_i \in \mathbb{R}^3, i = x, y, z$ can be calculated. The dot product with the highest absolute value indicates the associated perspective. The look-direction in turn is derived from the handheld's tracking pose.

3.4 Selection

3.4.1 Guidelines

For the design and evaluation of the selection technique, the principles from Section 3.2 can be extended to derive the following guidelines:

1. Keep direct touch abilities: *as stated in Section 3.2.*
2. Keep interaction simple: *as stated in Section 3.2.*
3. Enable disambiguation and unique selection: In dense virtual scenes, it is challenging to select objects that are partly or fully occluded, if they have a similar visual appearance (shape, color, texture). The selection technique should therefore display multiple selected objects in a way that provides insight into their original spatial arrangement to assist object disambiguation.

3.4.2 State-of-the-Art Selection Techniques

Though many 3D selection techniques are designed for AR scenarios, they can be adapted for handheld AR application as well. However, adjustments have to be reviewed carefully as they

could alter the techniques original characteristics so that a clean evaluation with a state-of-the-art (SOA) technique cannot be provided anymore. Due to the first requirement imposed from a handheld AR scenario, popular selection techniques for immersive VE that follow the *virtual hand* metaphor, as mentioned in Section 2.4.1, cannot be taken into account for the evaluation, as they require separate tracking of the user’s input and output device (e.g. a headtracker). Thus, a direct comparison is problematic or its adaption would flaw the evaluation, as mentioned before. The dual-finger selection techniques, described in Section 2.3.2, are not considered to be sufficient baseline techniques for the evaluation as they impose selection with multiple fingers and face problems near the screen borders. Selection by progressive refinement with *SQUAD*, as described in Section 2.3.3, fulfills the requirements of one-handed handheld AR but does not provide spatial clues to disambiguate visually similar objects as does its successor *Expand*.

Therefore, *Ray-casting* and *Expand* were found to be most appropriate as baseline techniques in the evaluation. Both fulfill the requirements from Section 3.1 and can easily be adapted to handheld AR without altering their original characteristics.

3.4.2.1 *Ray-casting*

As *Ray-casting* is a selection technique with a single interaction step, it can easily be adapted for one-handed handheld AR by utilizing a simple tap-gesture. The ray-cast is triggered by a single tap on the screen denoted as $p_T = (x, y) \in \mathbb{R}^2$. The handheld’s virtual position $P_H \in \mathbb{R}^3$, its orientation $O_H \in \mathbb{R}^4$ and the virtual camera’s field of view $fov = (fov_x, fov_y)$ are known parameters in the virtual environment. The ray-cast’s origin and direction $\vec{v}(P_H, \vec{d}) \in \mathbb{R}^3$ in the virtual scene can be provided with back-projection as follows:

1. Calculate the virtual camera’s frustum’s dimension F_D at an arbitrary depth z with $F_D = (F_w, F_h) = (2z \cdot \tan(\frac{fov_x}{2}), 2z \cdot \tan(\frac{fov_y}{2}))$, using the horizontal and vertical fov and the trigonometric tangent function.
2. Given that the screen coordinate system has its origin in the upper left corner, the normalised screen coordinates p_{Tn} can be derived from the touch point p_T . Using the device’s screen’s size $S = (s_w, s_h)$ the normalised touch point is calculated with $p_{Tn} = (x_n, y_n) = (\frac{x}{s_w}, \frac{y}{s_h}) - 0.5$. Normalised screen coordinates are used to map the real 2D touch point on the screen to a virtual point in 3D space.
3. The ray-cast is then defined with the vector $\vec{v}(P_H, \vec{d})$ from the handheld’s origin P_H into the direction $\vec{d} = O_H \cdot (F_w \cdot x_n, F_h \cdot y_n, z)$.

The first object that is casted by the virtual ray is selected and presented to the user. Selection by Ray-casting resembles therefore in a simple one-step interaction technique.

3.4.2.2 *Expand*

The two-step selection technique *Expand* can be adapted for one-handed handheld AR, too, solely using simple tap gestures. The technique features three phases of which two are interactive. The first phase is similar to Ray-casting. The selection is triggered with a single tap on the

device’s screen to indicate the direction of the *Cone-cast* [25]. All objects intersecting the cone will be subject of the next phases to refine the selection. The second phase is a non-interactive animation rendered from the virtual camera’s perspective upon selection. It illustrates the spatial transition of the objects from their original location in the virtual scene to their designated position on the grid. The AR scene is hidden upon completion of the animation. In the third phase, the objects are aligned on a grid in front of a solid gray background, so each object can be conveniently accessed. The refinement, and thereby the selection, ends with a second tap on the screen that uses Ray-casting (as described in 3.4.2.1) to select the desired object from the grid.

As the original publication [10] did not provide detailed information how the objects are aligned on the grid to reflect their original spatial arrangement, a detailed description of the handheld adaption is given in the following.

1. For n objects that intersect the Cone-cast, a dynamically sized grid is created offering $m \geq 4n$ cells to provide an adequate number of abstract positions to resemble their original setting. Each cell is represented by its center $c_i \in \mathbb{N}^2$ on the screen.
2. For each of the n objects, the corresponding 2D screen position $p_i \in \mathbb{N}^2$ is calculated.
3. Finally, for each screen point p_i , the closest cell-center c_i is determined and the object is placed at the cell’s center position.

Expand is designed to work for dense conditions when multiple objects may be subject of a selection. It provides two-dimensional spatial context preservation and precise selection of objects that are partly or completely occluded.

3.4.3 DrillSample Selection Technique

Derived from Ray-casting and Expand, the novel selection technique *DrillSample* is designed in an iterative way according to the outlined guidelines and to meet the necessary requirements. The selection process is organized in two steps and all touch interactions are performed with one finger. The technique especially supports disambiguation and selection of objects that are partly or fully occluded, even if they are of high visual similarity. The technique’s principle and name were inspired from taking a drill (core) sample for geological measures.

The selection starts with a single tap on the screen which triggers Ray-casting. Instead of immediately returning the first object that intersects the ray, all casted objects are collected defining the set of objects that are subject of the following refinement. In the second phase, all casted objects are cloned and displayed as if they were pulled out of the virtual scene, thus constituting a drill sample. The drill sample is aligned coplanar to the screen’s image plane in front of a solid gray background. The AR scene is not visible during refinement. On the very left of the drill sample, a 3D model of a smartphone depicts the origin of the ray-cast and on the very right, the most distant intersected object denotes its end (see Figure 3.6). The clones on the drill sample maintain their original arrangement of the context, their local hit-point with the ray-cast and the distance to each other, they were extracted from. Therefore, the DrillSample selection extends Expand’s idea to the depth domain and thus assists in object disambiguation

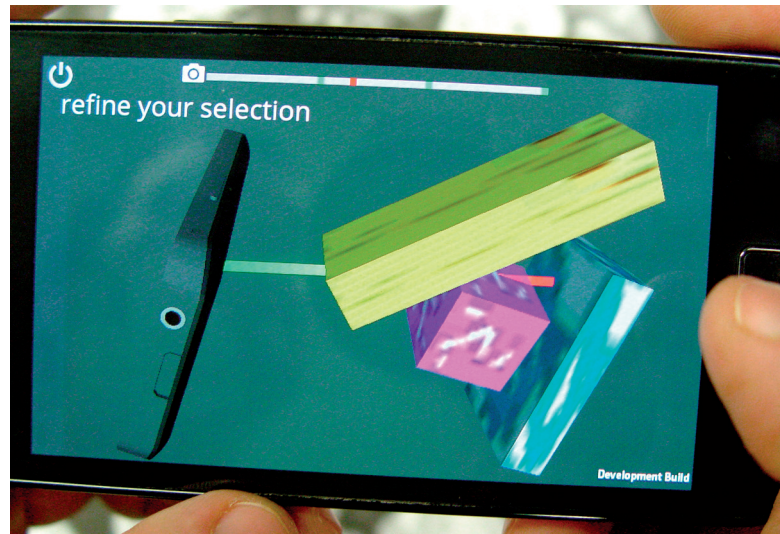


Figure 3.6: *DrillSample* visualisation on a smartphone.

and selection. The drill sample visualisation allows for a more detailed inspection by using the following interactions:

1. The handheld's built-in IMU¹ can be used to rotate the whole visualisation, so that the objects can be viewed from different perspectives.
2. With a horizontal one-finger swipe gesture, the objects can be browsed by traveling along the ray, bringing different objects to the center of the screen.
3. With a vertical one-finger swipe gesture (or optionally an undirected two-finger pinch gesture), the virtual camera can be zoomed in and out to provide either a more detailed look or a better overview.

The second phase ends with the selection of the designated object by a simple tap gesture. The user is informed about the selection of the object, the clones of the drill sample visualisation are destroyed and the AR scene is again displayed. For situations in which the ray-cast intersects only a single object, the drill sample visualisation is omitted and the object is selected immediately. The illustrations in Figure 3.7 depict the different interactions with the handheld device throughout the different phases. The diagram in Figure 3.8 shows all states of the *DrillSample* selection technique and the transitions in between according to the input of the user.

3.4.3.1 Algorithm

To formalise the illustrated selection process, the proposed *DrillSample* algorithm can be denoted as follows:

¹An inertial measurement unit (IMU) is an electronic sensor used to measure the current rate and 3D direction of acceleration as well as rotational changes along the pitch, roll and yaw axes.

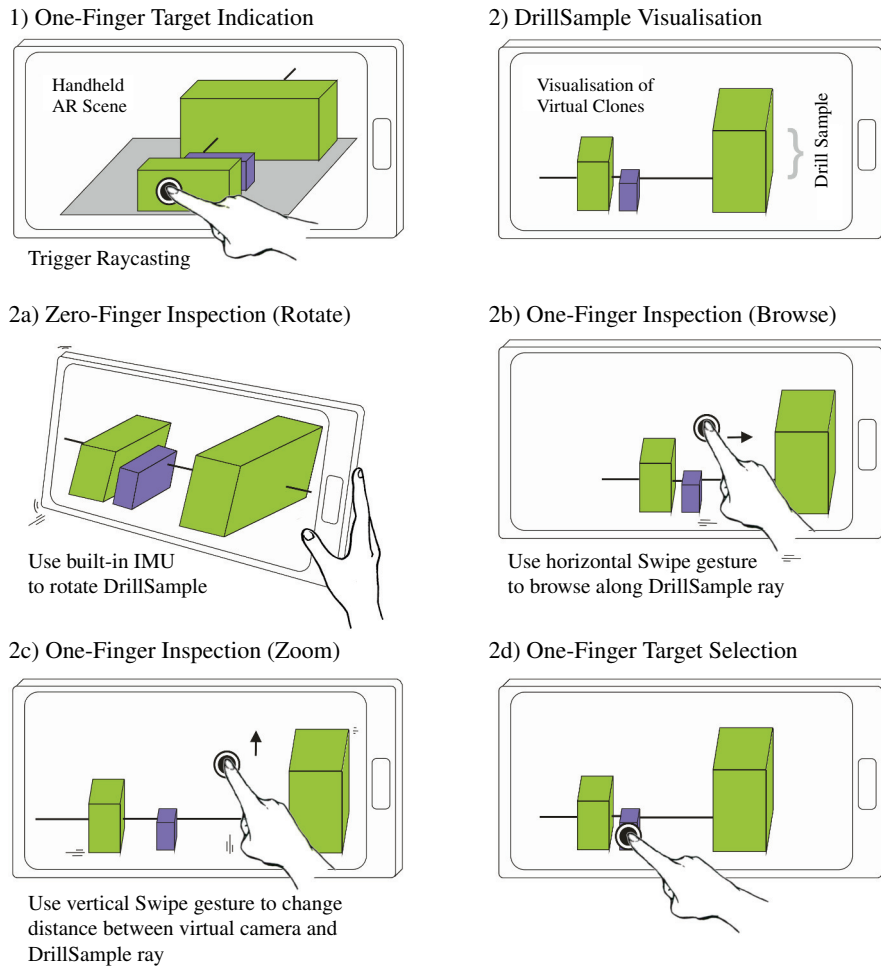


Figure 3.7: *DrillSample's* two-step selection process.

Step 1: Target indication & DrillSample construction

1. The user performs a touch on the device's screen at $p_T \in \mathbb{R}^2$.
2. Perform Ray-casting, as described in Section 3.4.2.1, but return all casted objects.
3. Create clones for each object, storing its original orientation, visual appearance and hit-point.
4. Optimise the length of the DrillSample (see Section 3.4.3.2).
5. Calculate a pivot point at the center of all hit-points.
6. Rotate the drill sample around the pivot point, so that a) the drill sample lies parallel to the image plane's horizontal axis and b) the object hit first is displayed on the left side of the screen (see Figure 3.6).

7. Z-Positioning of the drill sample (see Section 3.4.3.2).

Step 2: Optional inspections during refinement step

1. Calculate touch points and direction of the vertical swipe gesture (or optional two-finger pinch gesture) to change the distance between the virtual camera and the drill sample (see Section 3.4.3.2).
2. Calculate touch points and direction of the horizontal swipe gestures to travel along the drill sample if it spans multiple screens.
3. Rotate the drill sample by mapping the device's gyroscopic sensor values to the pivot point of step 1.5.

Step3: Final target selection

1. Select object by using single touch point's coordinates and Ray-casting, as described in Section 3.4.2.1.
2. Destroy clones and the drill sample visualisation and return to AR view.

3.4.3.2 Important Aspects

To provide a reasonably refined visualisation of the virtual clones, there are two important aspects:

1. The length of the drill sample ray needs to be optimised while preventing visual intersection of the clones and preserving their relative distances.
2. The optimal z-position of the drill sample to the virtual camera must be obtained.

Length of the drill sample Ray Since the relative distance of objects to each other is sufficient to preserve the spatial context, the real length of the ray should be scaled for its visualisation to provide an optimal overview. If the objects are far away from each other the ray might be shortened. To reveal objects that are inside of another, e.g. a ball in a bucket, it might have to be stretched. The optimal scaling amount of the ray depends on the shortest distance between the convex hulls of the two neighbouring objects along the drill sample ray. For objects with overlapping hulls, the distance is specified as a negative value, and positive otherwise. Assuming n objects on the drill sample and the shortest distance between $(n - 1)$ neighbours is denoted by d_i , the length of the ray is then modified by $-d_i \cdot (n - 1)$. The calculation of the precise distances can be computational expensive, especially in dense environments with complex shapes. To minimise the computational load, an approximation with linear complexity is used by treating all objects as spheres (see Figure 3.9) with the maximum extent of the objects' bounding box used as its radius and the hit point as its center. For objects whose center point is not close to the ray or have a complex concave shape, this may not be visually pleasing as it overestimates the

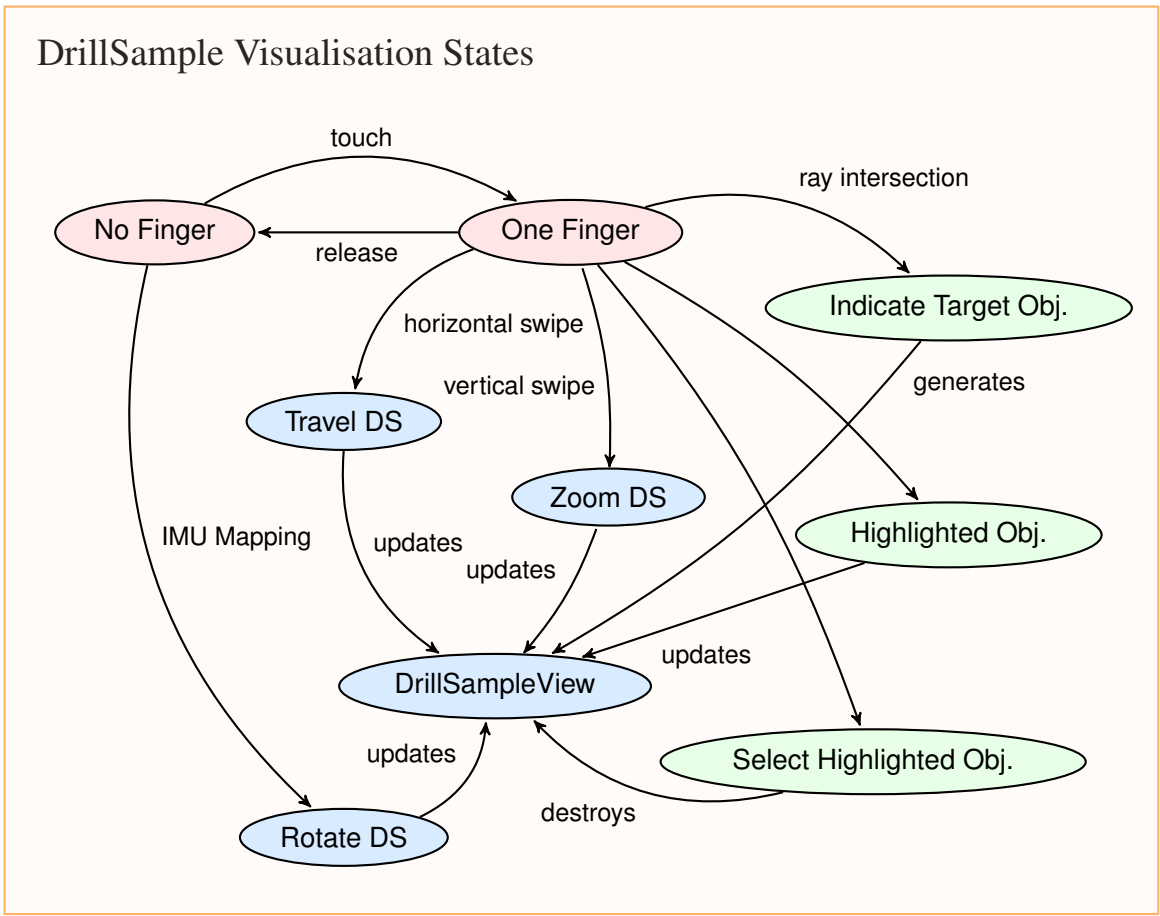


Figure 3.8: State diagram for the DrillSample selection technique.

real distance between neighbouring objects. More elaborate algorithms can be employed in the future to enable an optimal adjustment of the length.

Z-Position of the DrillSample In the current algorithm an overview of all selected objects is presented. This results in the following crucial aspects.

1. The more the distance between clones varies, the less the drill sample can be compressed. To provide an overview of all clones on a single screen, the drill sample must be positioned at greater distance to the virtual camera. This might result in small clones being barely visible.
2. The more the size of the clones varies, the less likely there is a distance to the virtual camera at which all clones are visible. Small objects may appear too small or large objects might be clipped at the near image plane.

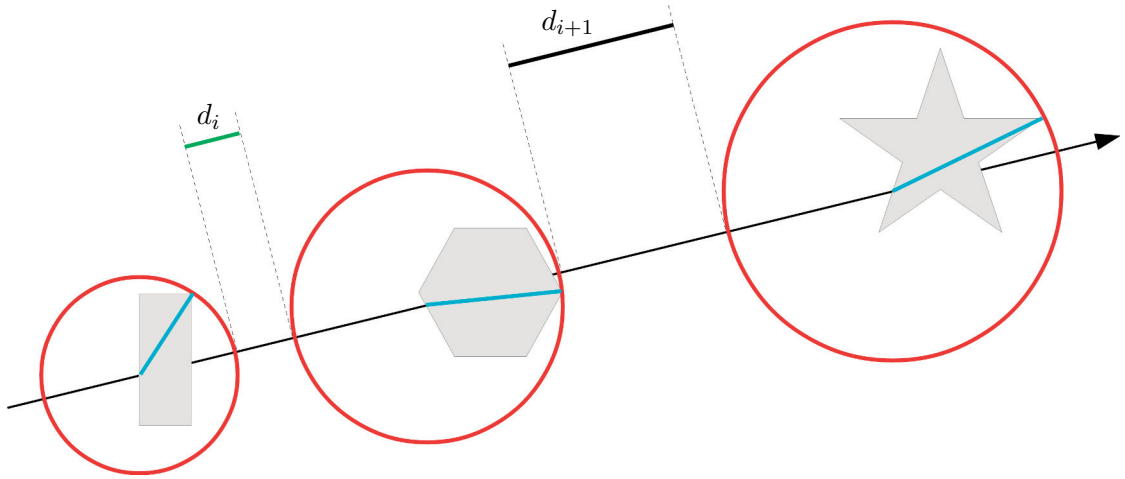


Figure 3.9: Sphere approximation of clones' size to calculate the optimal ray length.

3. The more objects are selected, the less likely the overview provides a meaningful starting point for refinement, as the clones in the overview appear too small as in critical aspect (1).

Thus, the distance between the virtual camera and the drill sample depends on the size of the clones and their relative distance to each other. To obtain an overview the distance D_{ov} , between the virtual camera and the drill sample, can be calculated as follows:

$$D_{ov}(B_{DSS}) = \frac{exp}{\tan(fov \cdot 0.5)} + B_{DSS}(z), \text{ where} \quad (3.1)$$

$$exp = \begin{cases} B_{DSS}(y), & \text{if } R_B < R_{fov} \\ B_{DSS}(x), & \text{otherwise} \end{cases} \quad (3.2)$$

$$fov = \begin{cases} fov(y), & \text{if } R_B < R_{fov} \\ fov(x), & \text{otherwise} \end{cases} \quad (3.3)$$

While $B_{DSS} \in \mathbb{R}^3$ is the DrillSample's axis-aligned bounding box represented as an expansion vector, R_{fov} is the aspect ratio of the virtual camera's field of view, R_B the aspect ratio of the bounding box's side facing the camera and fov is the field of view of the virtual camera. Additionally, it has to be ensured that neither the near, nor the far clipping plane of the virtual camera are violated. The interval, in which users may modify the distance of the camera to the drill sample with a vertical swipe gesture (see illustration 2c in Figure 3.7), is then limited to $[D_{ov}(B_C), D_{ov}(B_{DSS})]$ by the bounding box of the biggest clone B_C on the drill sample and by the bounding box of the drill sample B_{DSS} itself. However, the *optimal* distance depends on the specific application.

3.5 Manipulation

Once an object has been selected, independent of the selection technique involved, it can be manipulated. This section discusses the design of a manipulation technique suitable for one-handed handheld AR.

3.5.1 Guidelines

The guidelines that are applied for designing the manipulation techniques are the same as stated in Section 3.2.

1. Keep direct touch abilities.
2. Keep interaction simple.

3.5.2 State-of-the-Art Selection Manipulation Techniques

The manipulation techniques presented in Section 2.4 are not explicitly designed for one-handed handheld AR use cases. Their usage is therefore discussed in the following.

Techniques using the virtual hand metaphor [4] like go-go [36] or HOMER [5] enable for intuitive and effective 6DOF manipulation (rotation and translation). However, they are defined for (immersive) VEs and require separate tracking of the 6DOF pose of the input and output device. This directly conflicts with the *single I/O device* requirement imposed by the handheld setup and can not be used without further adaption. Touch-based interaction techniques like the Z-Technique [28] or the Dual-Finger techniques [51] use DOF-separation to decompose 6DOF manipulations (e.g. translation and rotation) into 3DOF tasks and these in turn into subtasks with 1DOF and 2DOF. The subtasks are then performed with specific multi-touch gestures. Telkenaroglu [51] defines manipulation techniques for all three tasks, but uses inconsistent gestures that might cause confusion as described in Section 2.4.3. The Z-Technique is only defined for translation but not rotation and scaling. While Martinet [28] and Telkenaroglu [51] use DOF-separation to handle 3D manipulation, Reisman [40] instead integrates 6DOF by employing two-handed three-finger gestures to extend RST (Rotate-Scale-Translate) operations from 2D to 3D. However, the use of multi-hand input violates the requirement of *limited gesture complexity*.

Since the state-of-the-art manipulation techniques cannot be employed in one-handed handheld AR scenarios, two novel techniques are proposed. The technique *3D Touch* focuses on touch input and consistent DOF separation whilst with *HOMER-S*, the emphasis is put on integral manipulation up to 6DOF avoiding touch input at all.

3.5.3 3D Touch

Recent work of Veit [52] suggests that the integration of DOF does not necessarily improve the performance for orientation tasks. This complies with findings of Martinet [29] revealing that DOF integration for rotation and translation reduces not only performance but also satisfaction. Furthermore, they suggest that an interaction technique should rather follow the structure of the

input device than the structure of the specific task. This inevitably leads to DOF separation using a 2D touch interface for 3D manipulation tasks.

According to Veit’s work, integral 3D transformations are separated into 3DOF entities to rotate, scale and translate. To comply with the requirement of limited gesture complexity, each entity is further split into 1-2DOF manipulation tasks by employing the current device pose of the handheld that is provided through optical tracking. Inspired by [40], the 2D touch coordinate in screen-space are transformed to 3D space to support the guideline in order to keep direct touch abilities and to achieve non-uniform manipulations. The proposed manipulation technique 3D Touch manages to rely solely on familiar one- (translate, rotate) or two-finger (scale) gestures.

3.5.3.1 Degree of Freedom Limitation

The key feature of 3D Touch is the seamless transition between the various 1-2DOF subtasks of a 3DOF manipulation entity. This allows users to switch easily between the subtasks without using an abstract mode switch (e.g. like a button) or by applying a distinct gesture for each subtask. Changing the viewpoint and perspective is an elemental form of interaction in handheld AR setups. Therefore, the pose of the handheld device and the resulting perspective of the object to manipulate is used to identify which DOFs are currently accessible. Depending on the manipulation task, but also on the user’s needs, a manipulation can be performed either with respect to the world coordinate system or to the local coordinate system of the object to manipulate. Therefore 3D touch allows users to switch between manipulation in local and global space.

The simultaneous accessible 1-2DOF subtasks are determined by measuring the collinearity of the unit vectors of the targeted (global or local) coordinate system with the look-direction \vec{d}_H in world space of the handheld device. \vec{d}_H can be derived from the handhelds orientation $O_H \in \mathbb{R}^4$ by transforming the local look-direction to world space. If the targeted coordinate system’s unit vectors $\vec{e}_i \in \mathbb{R}^3, i = \{x, y, z\}$ are not already in world space, they need to be transformed to world space as well. Given that \vec{e}_i and \vec{d}_H are normalised, a measure $c_i \in [0, 1]$ for each unit vector \vec{e}_i can be calculated with the dot product by $c_i = |\vec{e}_i \cdot \vec{d}_H|$. The smaller c_i is, the better is the dimension accessible from the touch screen. A good empirical threshold to disallow manipulation of a subtasks was found at $c_i > 0.72$.

To support the user’s decisions while performing the manipulation, the c_i -values can be used to indicate which dimensions are accessible from the current perspective. However, which dimensions are actually accessed is determined once one-finger input is used on the touch screen. 3D Touch features a strict 1DOF and a mixed 1-2DOF manipulation mode that are explained as follows.

Strict One Degree Of Freedom Manipulation This mode allows manipulation of only 1DOF at a time. To perform a manipulation in strict 1DOF mode, users would perform a one-finger drag gesture (see Section 2.5) along one of the dimensions that are indicated to be accessible. The dimension to alter and its extent is then identified by evaluating the drag gesture for collinearity with the unit vectors of the targeted coordinate system as follows.

1. Obtain the 3D points $P_t, P_{t+1} \in \mathbb{R}^2$ in world space for two successive touch points $p_t, p_{t+1} \in \mathbb{R}^2$ of the drag gesture on the touchscreen for the current $t + 1$ and the last t frame. The 3D points are calculated using back-projection (see Section 3.4.2.1) at the distance $z = \|P_H - P_O\|$ of the handheld's position P_H and the position of the object to manipulate P_O .
2. Calculate a measure of collinearity $k_i \in [0, 1]$ for each accessible dimension using the dimensions' corresponding unit vector $\vec{e}_i \in \mathbb{R}^3, i = \{x, y, z\}$ with $k_i = \vec{e}_i \cdot (P_{t+1} - P_t)$. Then, the highest absolute value $|k_i|$ determines the dimension \vec{e}_i along which the manipulation is applied. Furthermore, the direction of the manipulation is defined with the sign $s = \frac{k_i}{|k_i|}$.
3. Calculate the extent a of the manipulation with $a = \|P_{t+1} - P_t\|$ as the absolute distance between the two back-projected touch points. As the back-projection is calculated in respect to the distance between the handheld and the manipulated object in step 1, the extend a scales non-uniformly.

Manipulations for a single dimension are therefore expressed with the 3-tuple $M_L = (\vec{e}_i, s, a)$.

Mixed 1-2 Degree Of Freedom Manipulation This mode allows simultaneous 2DOF manipulation at a time, but only if the image plane is almost coplanar with two dimensions of the targeted coordinate system. Otherwise, it switches back to the strict 1DOF manipulation mode as described previously. The mixed 1-2DOF manipulation mode operates as follows.

1. Check for near coplanarity of the handheld's image plane with any two dimensions of the targeted coordinate system with $(c_i > 0.9) \wedge (c_j > 0.9) \wedge (i \neq j)$, with $i, j \in \{x, y, z\}$. If this condition is satisfied operate in 2DOF mode and continue with step 2, otherwise switch to strict 1DOF mode.
2. Obtain the 3D points $P_t, P_{t+1} \in \mathbb{R}^2$ of the touch input using back-projection (see step 1 of strict 1DOF mode).
3. Calculate measures of collinearity k_i, k_j of the touch input for the two accessible dimensions \vec{e}_i and \vec{e}_j (see step 2 of strict 1DOF mode).
4. Calculate the manipulation's extent with $a = \|P_{t+1} - P_t\|$ (step 3 of strict 1DOF mode).
5. Allocate the dimension specific extents a_i, a_j of the manipulation, so that the extent a is distributed according to the level the dimensions' colinearity with $a_i = a \cdot \frac{k_i}{|k_i| + |k_j|}$ and $a_j = a \cdot \frac{k_j}{|k_i| + |k_j|}$.

Manipulations on a plane are therefore expressed with the 4-tuple $M_P = (\vec{e}_i, \vec{e}_j, a_i, a_j)$.

3.5.3.2 Translation

3D translations are performed using single touch input for the current and last frame $p_t, p_{t+1} \in \mathbb{R}^2$ that are combined with the current device pose, as described in Section 3.5.3.1. Depending on the manipulation mode and the perspective, the manipulation is either defined with (\vec{e}_i, s, a) for 1DOF or $(\vec{e}_i, \vec{e}_j, a_i, a_j)$ for 2DOF manipulations. Given the objects position P_O in virtual space, the object's manipulated position P'_O can be calculated with:

$$P'_O = P_O + (s \cdot a \cdot \vec{e}_i) \quad \text{or}$$

$$P'_O = P_O + (a_i \cdot \vec{e}_i + a_j \cdot \vec{e}_j)$$

Subsequently, the 3D position of the selected object is adjusted. In Figure 3.10, some examples of the resulting 3D translations using the proposed 3DTouch algorithm are illustrated. Moving the finger right or left in Figure 3.10(a) causes a translation along the x-axis. Analogously, moving the finger up and down in Figure 3.10(b), results in translations along the y-axis, or z-axis, respectively (Figure 3.10(c)).

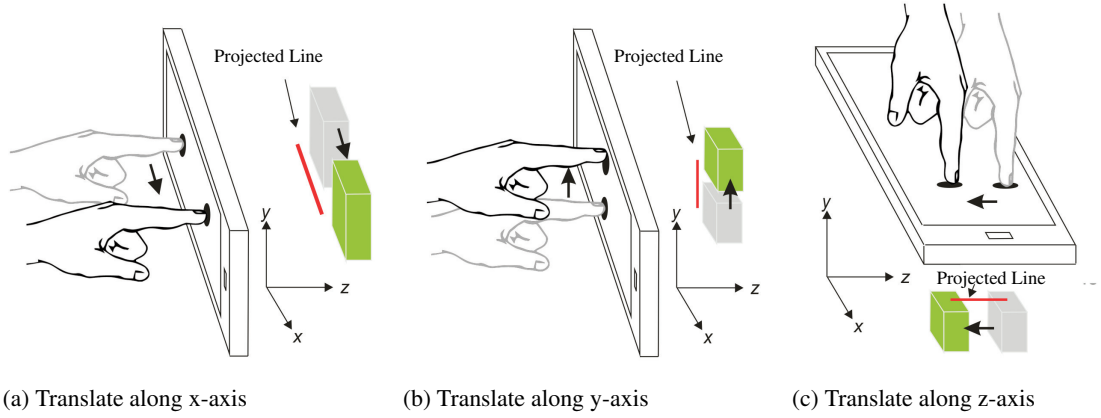


Figure 3.10: Examples of translations using 3DTouch.

3.5.3.3 Rotation

Similar to translations, 3D Rotations are performed using single touch and the device pose. The algorithm is based on the proposed translation algorithm in Section 3.5.3.1. However, before back projecting the two touch points, a line perpendicular to $line(p_t, p_{t+1}) \in \mathbb{R}^2$ is calculated and subsequently the two new points are back projected. The angle of rotation $F_R \in \mathbb{R}^3$ is then calculated with:

$$F_R = \frac{360 \cdot s \cdot a \cdot \vec{e}_i}{U} \quad \text{or}$$

$$F_R = \frac{(360 \cdot a_i \cdot \vec{e}_i) + (360 \cdot a_j \cdot \vec{e}_j)}{U} ,$$

depending on the manipulation mode and the perspective the object is examined from. The scalars a or a_i, a_j regulate the angle as a fraction of the circumference U of the bounding sphere of the manipulated object. The factor F_R is then applied to the current rotation in the object's local coordinate system. In Figure 3.11, examples of the resulting 3D rotations using the proposed 3D Touch algorithm are illustrated. Moving the finger right and left in Figure 3.11(a) causes a rotation around the x-axis, or y-axis, respectively (Figure 3.11(b)). Analogously, moving the finger back and forth in Figure 3.11(c) results in a rotation around the z-axis. While translation and rotation in 3D are performed using single touch, 3D Touch scales objects using two-finger touch input.

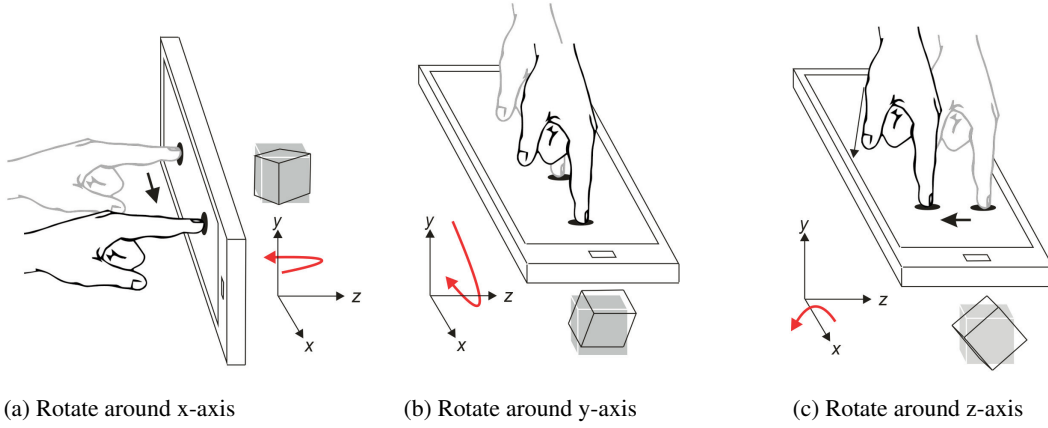


Figure 3.11: Examples of rotations using 3D Touch.

3.5.3.4 Scaling

The proposed scaling algorithm supports non-uniform scaling along each axis. Therefore, a two finger pinch-like gesture and an adapted version of the proposed algorithm from Section 3.5.3.1 are applied. Both of the two touch pairs $p_{T_1}, p_{T_2} \in \mathbb{R}^2$ of the current frame ($t + 1$) and the previous t are back projected into 3D, resulting in two manipulations tuples M_L for 1DOF or M_P for 2DOF manipulations. Scaling depends on whether a user moves both fingers together or apart and on the magnitude of that movement. The scaling factor $F_S \in \mathbb{R}^3$ is then calculated with:

$$F_S = (a_{t+1} - a_t)\vec{e}_{i,t} \quad \text{or} \\ F_S = (|a_{i,t+1}| - |a_{i,t}|)\vec{e}_{i,t} + (|a_{j,t+1}| - |a_{j,t}|)\vec{e}_{j,t}$$

For positive scaling, F_S is bigger than 0, for negative, the value is smaller than 0. F_S is then added to the current scale in the object's local coordinate system.

3.5.4 HOMER-S

3D Touch is straightforward and simple. However, the touch abstraction layer still exists and manipulation is limited to the screen size of the handheld device. Therefore, the novel HOMER-S technique is introduced, which integrates all 6DOF of a translation and rotation task by directly mapping the handheld's pose onto the object. Scaling is separated as a 3DOF task by reusing the device's position information. Thereby, real-world metaphors are imitated for translation, rotation and scaling to eliminate touch input during manipulation and to extend the interaction space to the user's physical space.

Inspired by Henrysson [15] and using the immersive 3D manipulation technique HOMER as a baseline technique, HOMER is adapted to use it on smartphones and tablets (therefore HOMER-S). The original algorithm uses the 6DOF pose of the user's torso and that of the interaction device to manipulate an object. Since a handheld AR environment has different characteristics than an immersive VE, HOMER is adapted, as illustrated in Figure 3.12.

Rotations of the selected object around arbitrary axes are independently controlled. An isomorphic mapping between the handheld's orientation and the virtual hand is applied to rotate an object around the hit point used as pivot point. Thereby, the physical movement and rotation of the handheld device directly influences the transformation of the selected object. By performing Ray-casting, the object is released and the virtual hand moves back to the handheld's position. The proposed HOMER-S algorithm is denoted as follows:

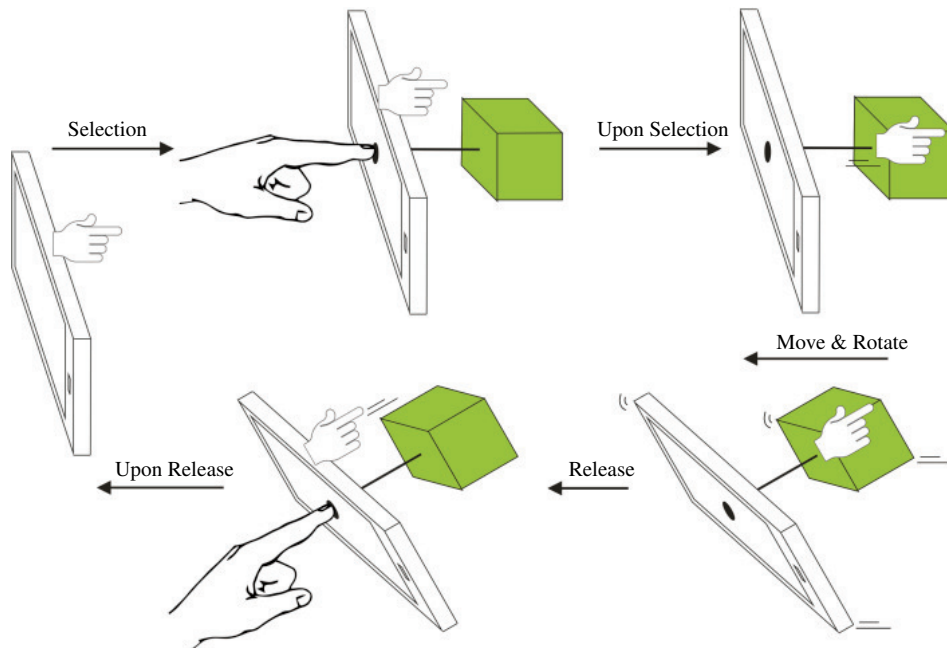


Figure 3.12: 6DOF manipulation using HOMER-S.

1. While no object O with position Op and orientation Oo is selected, read the handheld's position hp and orientation ho and map it to virtual hand's position, so that:

$$vh_p = h_p, vh_o = h_o$$

2. Upon selection, store current handheld's position $vh_{sel} = h_p$.
3. At each frame:
 - Perform rotation, set $O_o = h_o$
 - Perform translation,
 - Set $vh_{curr} = h_p$
 - Calculate distance $d(vh_{curr}, vh_{sel})$
 - Normalise vector $\vec{v}(vh_{sel}, vh_{curr})$
 - Set $vh_p = vh_{sel} + d \cdot \vec{v}_{norm}$

To scale an object, the position information of the virtual hand is re-used. Therefore, at each frame the difference between the 3D positions of the previous and the current frame is calculated. The delta information is subsequently mapped onto the selected object. Thus, moving the virtual hand in positive direction of each axis will scale up; moving in negative will scale down the object. Thereby, non-uniform scaling along all axes can be achieved. As described, HOMER-S provides touchless manipulation integrating all 6DOF for simultaneous translation and rotation as well as 3D scaling. Thereby, the touch abstraction layer that requires prior knowledge is replaced with real-world metaphors. Additionally, the user's interaction space is no longer limited to physical screen size, but is expanded dramatically.

3.5.5 Mode Switches

As 3D Touch offers RST-transformations (Rotate-Scale-Transform) by decomposing each transformation into a single 3DOF entity, mode switches are required. Compared to previous work from Reisman [40], Telkenaroglu and Capin [51], an additional extra input modality is introduced. However, as reported in literature by Martinet [29] and Veit [52], separating the DOF of the task leads to better results, than trying to use the separated DOF of a multi-touch display in an integral way, as demonstrated in [40].

For HOMER-S, no degree-of-freedom separation needs to be performed, since it provides translation and rotation in an integral way. But if desired, translation and rotation can also be separated into single transformation entities. HOMER-S takes advantage of re-using real world metaphors to provide intuitive translation, rotation and scale. However, the metaphors for translating and scaling are akin in movement and hence are hard to distinguish when only using the device pose. Thus, conscious mode-switching between translate/rotate and scale is used rather than introducing a new and hence probably unknown real-world metaphor to handle scaling.

To summarise, 3D Touch provides the 3D transformations (1) *translate*, (2) *rotate* and (3) *scale*. HOMER-S provides additionally the transformation (4) *translate & rotate*, which make (1) and (2) optional. As illustrated in Figure 3.13, the mode switches are realised with a simple button interface that appears upon selection.

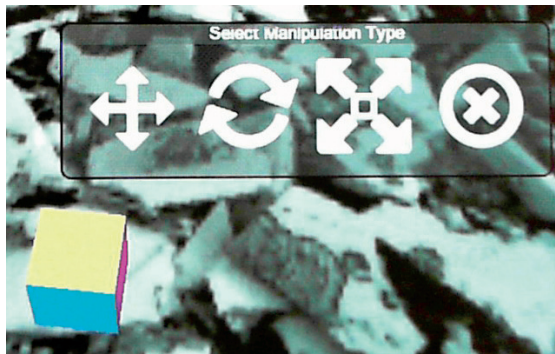


Figure 3.13: GUI of 3D Touch’s manipulation types upon selection.

3.5.6 Important Aspects

The proposed manipulation techniques offer intuitive handling of 6DOF manipulation, but introduce some important aspects as well. Since both methods are designed for handheld AR environments, a valid device pose is required. Thus, loss of tracking results in malfunction of object manipulation. Nevertheless, since there are several tracking methods for handheld devices available, a complete loss of tracking can be eliminated by combining two or more tracking techniques. Future implementations could overcome the loss of natural feature tracking by using the built-in inertial unit for pose estimation for a short period of time. Freezing the AR view as in [24] or [14] to stabilise the device pose during interactions with HOMER-S was neglected to allow an uninterrupted immersion.

Regarding rotations with HOMER-S, a drawback is caused by the direct mapping of the device orientation onto selected objects. Given the implicit binding of the input- and output device in handheld AR, rotations around the pitch axis are limited. This is especially the case, if only one marker for natural feature tracking is used, since the system can lose tracking quickly. 360° rotations around the yaw axis can be applied by real world movements of the user, while rotations around the roll axis are straightforward and adapt the steering wheel metaphor. Further research will be conducted to evaluate the usability of a non 1:1 mapping to compensate rotation drawbacks of pitch and yaw rotations.

Implementation

This chapter documents important implementational effort that incurred working on this thesis. A description of high level tasks that need to be fulfilled are documented in detail in the following sections.

4.1 System Design

The interaction techniques designed for modeling, selection and manipulation, described in Chapter 3, as well as the tools to conduct the user study are implemented in different applications. This section provides an overview of the applications that were developed as part of this thesis together with the involved hardware and software components.

4.1.1 Hardware

All applications created are targeted for handheld devices running an Android 4.x operating system. The hardware used for testing and deployment is a Samsung Galaxy S II I9100. The smartphone features a 4.3" WVGA capacitive multi-touchscreen, an Arm Cortex A9 Dual Core-Processor @ 1.2GHz, 1GB of main memory and a 8 mega pixel camera. It weights 116g and has the physical dimensions of 125.3 x 66.1 x 8.49 mm. For development, a standard laptop running Microsoft Windows 7 Professional 64bit with 16GB main memory, a 3.2GHz Intel i3210M dual-core processor and a NVidia GeForce GT620M graphics adapter came to use.

The image registration and tracking for the the augmented reality environment is realised by using the Vuforia AR SDK v2.8.7 from Qualcomm [37]. The development kit uses methods of natural feature tracking of predefined passive optical markers. For testing the marker was printed on A4 whilst for the user study it was printed on A2 (see Figure 4.1).



Figure 4.1: The marker used for optical tracking.

4.1.2 System Overview

The prototype developed as part of this thesis consists of three applications with a graphical user interface and one service running in the background (see Figure 4.2). The *UserStudy App* and the *Modeling App* are built with the Unity Game Engine [50] and use the Vuforia [37] framework to create an augmented reality experience. The *Storage Service* and the *Shape Detection App* are built with the standard Android SDK v4.4.2 [17] and the OpenCV library [34].

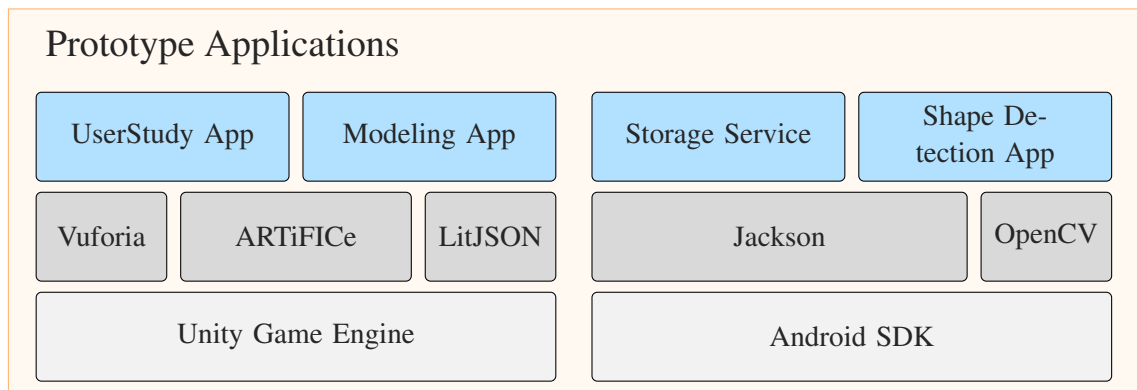


Figure 4.2: Prototype applications, 3rd party frameworks and development environment.

4.1.3 Applications

UserStudy Application The application’s purpose is to evaluate the novel selection technique *DrillSample* as well as the manipulation techniques *3DTouch* and *HOMER-S* in an experimental user study in a handheld AR setup. The application guides a participant through the experiment and collects data relevant to the study. A thorough description follows in Section 4.6.

Modeling Application Is the prototype of a handheld AR modeling application providing *Pre-fab Based Modeling* (see Section 3.3.2) of hand-drawn shapes with extrusion or lathe modeling techniques. Additionally, it implements mechanisms to synchronise user generated models in distributed, collaborative environments. This application is used in combination with the *Shape Detection Application* and the *Storage Service*. Details for this app's UI follow in Section 4.2.3. Section 4.3 covers the distribution of user generated content.

Shape Detection Application This application is also a part of the modeling prototype but is used by the *Modeling Application* to capture hand-drawn shapes with image processing algorithms. It implements a simple UI to capture shapes using the *view-to-find* analogy presented in Section 3.3.2.1 and a robust pipeline for shape detection with the OpenCV framework. To forward the captured shapes to the *Modeling Application*, it uses the *Storage Service*. Details about the user interface are described in Section 4.2.1 and the pipeline for shape detection is presented in Section 4.2.2.

Storage Service Is the background service acting as a communication agent between the *Modeling Application* and the *Shape Detection Application*. Its purpose is to enable inter-process communication for applications written in different programming languages (C# and Java). During the development of the service, the *JSONRMI* framework was developed to realise cross language IPC using networking. Details are given in Section 4.4.

4.1.4 Frameworks

This Section lists and describes all 3rd party frameworks that were used to create the different applications of the prototype.

Unity A game engine and development environment for interactive 2D and 3D applications [45]. It supports the programming languages C#, JavaScript and Boo. It is available for Mac OS X and Microsoft Windows platforms for development but can build executables additionally for Linux, PlayStation 3, Xbox, Wii, IOS, Android and PlayStation Vita. Unity is available as a free version with limited functionality or as a *pro* version with all features for the payment of license fees [50].

ARTiFICe A mixed reality framework developed at the Vienna University of Technology by the Interactive Media Systems Group. The framework builds around the Unity game engine and has a focus on the rapid integration of 3D input hardware, the development of interaction techniques and collaborative applications [31].

Vuforia AR SDK A software development kit developed by Qualcomm to build augmented reality applications for handheld devices running iOS or Android. The SDK uses optical inside-out tracking of a printed marker that is used for the registration of the virtual environment in real-world space. Vuforia is shipped with an extension for Unity and can therefore easily be used to create feature rich AR applications for handheld devices [37].

OpenCV A programming library providing advanced algorithms for computer vision and image processing. The library is written in C and C++ and provides various additional language bindings (e.g. Java, Python and Matlab). The library runs on Windows, OS X, Linux, iOS and Android and is distributed under the BSD license [8, 34].

LitJSON A library to process data formatted in JavaScript Object Notation [46]. The library is compatible with all .Net frameworks as well as Mono 2.0, so that it is fully compatible with Unity and all target platforms. The library is released into the public domain [3].

Jackson A data parser for formats like JSON, XML or CSV for the programming language Java. The library is dual licensed under Apache License (AL) 2.0 and LGPL 2.1.n [11].

4.1.5 Tasks & Overview

A prototype is developed based on the *ARTiFICe* framework and the *Unity3D* game engine to investigate and to design interaction techniques for *modeling*, *selection* and *manipulation* of virtual objects in one-handed handheld AR environments, as stated in Section 1.3. Additionally, a user study needs to be designed and executed to evaluate the newly created interaction techniques. The high level tasks part of this thesis are listed as follows.

- **Pen & Paper Shape Detection:** Develop a module that uses the handheld's back facing camera to detect shapes drawn on paper with the help of computer vision algorithms. The *Shape Detection Application* allows to select and save captured shapes to be used for polygonal modeling. As the computer vision algorithms will cause a high computational load, they need to be executed in a separate thread so that the user interface remains responsive.
- **Modeling Interactions:** Implement interaction techniques to generate a 3D polygonal mesh from the pre-captured 2D shapes using extrusion or lathe. Algorithms for polygon triangulation and mesh generation need to be implemented. The modeling interactions and the related algorithms are implemented in the *Modeling Application*.
- **User Generated Content Distribution:** The *ARTiFICe* framework can be used to build distributed collaborative AR applications but lacks functionality to synchronise user generated 3D content, such as polygonal models created with the *Modeling Application*. To enable state synchronisation for user generated content, *ARTiFICe* needs a module to distribute content amongst the participating users. The distribution layer should focus on sending as little data as possible to minimise network traffic.
- **Inter Process Communication:** The *Modeling Application* uses the *Shape Detection Application* to capture 2D shapes for the *Prefab Based Modeling* techniques (presented in Section 3.3.2). Therefore, it is required to implement a way of inter-process communication (IPC), to transfer the captured shapes from the shape detection to the modeling application. Furthermore, using a cross platform IPC, will also improve the development

process as deployment and debugging can be cumbersome with *Unity3D* and the *Vuforia* platform for handheld devices.

- **Selection, Manipulation Interactions:** The interaction techniques for selection (Drill-Sample) and manipulation (3DTouch and HOMER-S) of virtual objects in handheld augmented reality environments need to be designed and implemented. Along with state-of-the-art approaches that are used as baseline techniques in a userstudy, the novel interaction techniques will be integrated in the *ARTiFICe* framework. Existing interfaces of the framework are to be extended or altered if required.
- **UserStudy Application Design & Implementation:** The user study will employ a practical test in which users are asked to solve certain tasks with the competing interaction techniques. The test application will guide the users through the defined tasks, randomise the test set and record relevant data for the study.

4.2 Modeling

The polygonal modeling interface, as suggested in Section 3.3, can be divided in two steps. First, the detection of hand-drawn shapes with the help image processing, and second, the 3D polygonal modeling of the previous detected 2D shapes in the AR environment. Both steps require access to the handheld's camera. Shape detection grabs frames from the video feed to search for shapes in the image data, whilst the modeling interface requires the handheld's device pose in the AR environment provided by the optical tracking system. A concurrent execution of shape detection and modeling is impractical as they have different use cases involving the handheld's camera. The implementation of the modeling interface is therefore implemented in two applications.

The interface for the detection of hand-drawn shapes is designed to allow only little interaction in order to simplify its usage. The push-to-find analogy, as described in Section 3.3, supports the approach. The major difficulty of the implementation concerns the development of a performant and robust image processing pipeline to detect the hand-drawn shapes.

4.2.1 Shape Detection User Interface

The user interface features four buttons that are placed in the four corners of the handheld's screen so that they are easily accessible, but also block only little of the video feed displayed in full screen. Figure 4.3 shows a screenshot of the UI when shapes have already been detected and selected. Most important is the button B_1 which triggers the image processing thread to detect shapes with the *push-to-find* interaction. While B_1 is pressed, shapes are detected and the video feed is updated. The video feed a) is displayed in color while detecting shapes. Once detection has stopped, the video feed is displayed in grayscale and paused. Detected shapes are displayed in orange c) and turn light green b) when they are selected. Selected shapes can either be saved by double tapping or by pressing B_3 . Pressing B_2 saves the selected shapes as well, but also switches immediately back to AR mode for modeling. Button B_4 is used to display

the applications preferences so that parameters of the image processing pipeline, like e.g. the camera's preview resolution, are adjustable.

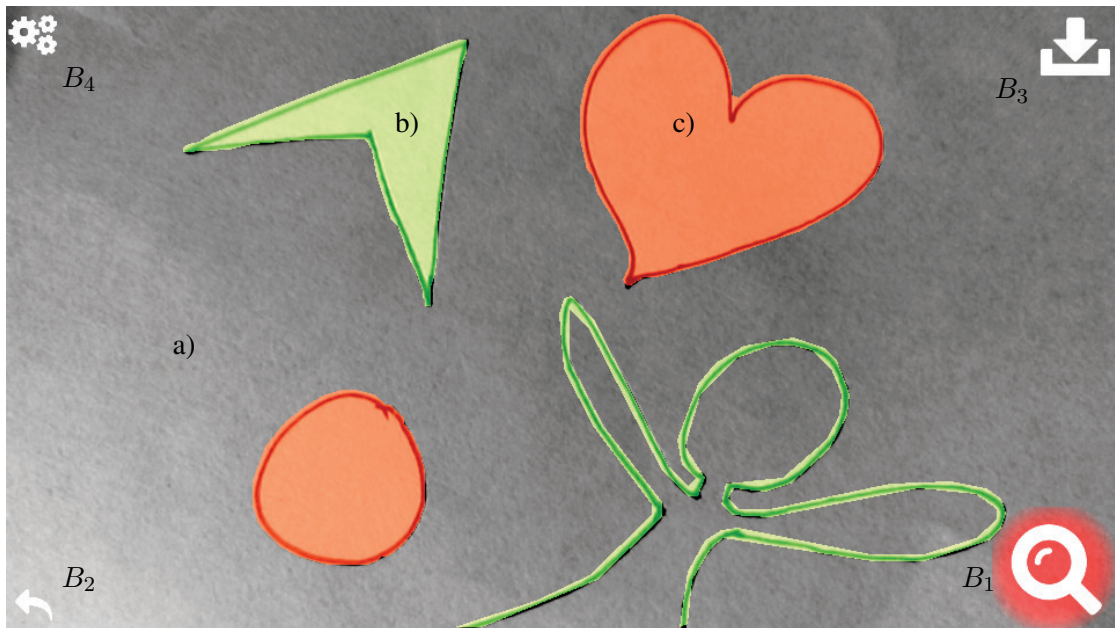


Figure 4.3: Shape detection user interface.

4.2.2 Image Processing

To maintain a responsive user experience, the pipeline runs on a separate thread in the background and is only called in case a new frame needs to be processed for shape detection. In return, frames that arrive for processing whilst a previous frame's processing has not finished yet, are discarded. The pipeline is implemented using the OpenCV [34] library and aims to reduce data as early as possible to minimise the computation time of subsequent steps. The detailed steps of the pipeline are displayed in Figure 4.4. Its states and critical parameters are addressed in the following enumeration.

1. **Preprocessing:** It is recommended to request a video feed from the device that coincides with the handheld's display resolution. This ensures that a detailed preview is provided for the user, but also avoids that unnecessary large frames are processed. Color information is not processed explicitly, each camera frame is therefore converted to a grayscale image. The conversion is done with OpenCV's conversion function `Imgproc.cvtColor(...)`.
In: Raw image (YUV) from the camera's video feed
Out: Grayscale image
2. **Downsampling:** Depending on the device's processor it might be necessary to further reduce the amount of data to achieve a short execution time. It is preferred to request a lower resolution feed from the camera directly, as this will not introduce an additional

Shape Detection Pipeline

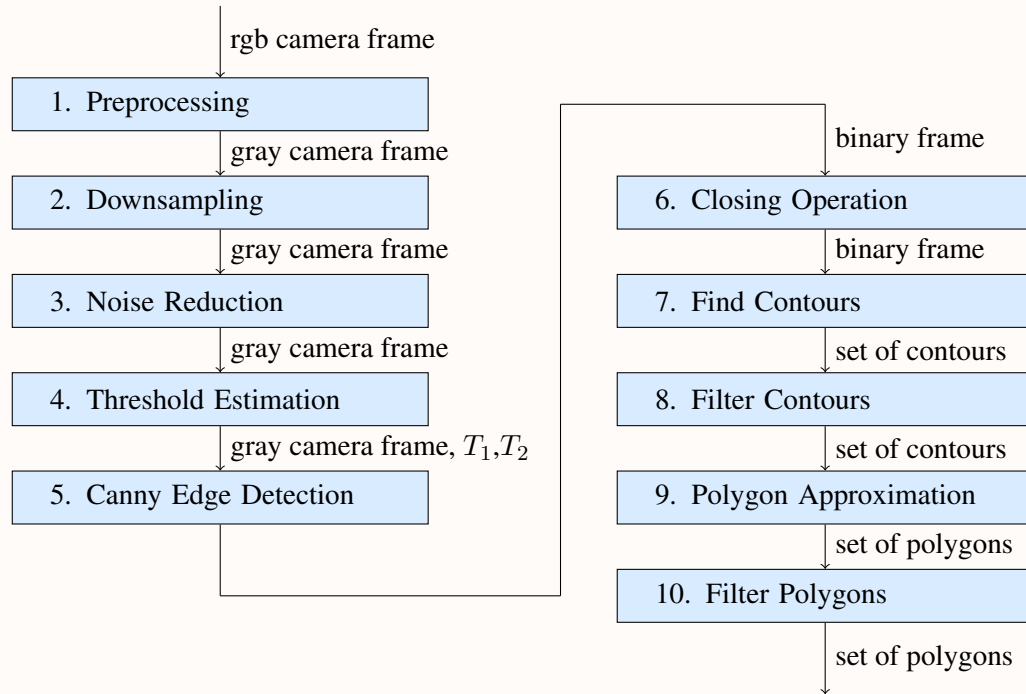


Figure 4.4: Image processing pipeline for robust shape detection.

computational load. However, using a low resolution for the video feed will look unappealing for users, as the presented feed lacks of details and sharpness. The computational load introduced by downsampling is, especially for higher resolutions, lower than for the upcoming image processing tasks. Downsampling can therefore help to make a compromise between computational load and an appealing presentation. Downsampling is done by using a bilinear interpolation with the function *Imgproc.resize(...)*.

In: Grayscale image

Out: Grayscale image

- Noise Reduction:** After the image has been downsampled, a 3×3 normalised box filter [49] is applied to reduce noise with the function *Imgproc.blur(...)*. Removing noise from the image will reduce the number of false detections of the Canny edge detector [9] and reduce the data being processed in the following steps.

In: Grayscale image

Out: Grayscale image

- Threshold Estimation:** To detect the edges of the hand-drawn shapes, the pipeline uses the Canny [9] edge detection algorithm. The quality of its result is mainly driven by its

hysteresis thresholds T_1 and T_2 . The two thresholds are therefore estimated for each frame as follows:

- a) Calculate a histogram with 32 bins.
- b) Determine the interval in which values are assigned.
- c) Find indices i_b, i_w of the strongest left- and righthanded bin from the center of the interval.
- d) Calculate intensity difference $I_d = \frac{255(i_w - i_b + 0.5)}{32}$.
- e) Estimate $T_1 = 0.2I_d$ and $T_2 = 0.8I_d$.

In words, the estimator searches for the strongest peak within dark and bright values. The intensity difference is then directly used to calculate a high entry threshold T_2 but a low exit threshold T_1 . This helps to skip small low intensities being detected and avoids that edges with intensity fluctuations being separated.

In: Grayscale image

Out: Canny thresholds T_1, T_2

5. **Canny Edge Detection:** Using the denoised grayscale image and the estimated Canny thresholds, Canny edge detection is performed using the function `Imgproc.Canny(...)`.
In: Grayscale image, T_1, T_2
Out: Binary image
6. **Closing Operation:** To further reduce noise and data, a morphological *closing operation* is applied to the binary image using a 3x3 pixel sized kernel. This will remove small structures from the image data and causes structures that have small holes to merge [49].
In: Binary image
Out: Binary image
7. **Find Contours:** The function `Imgproc.findContours(...)` provided by OpenCV can be used to detect contours within a binary image using a border following algorithm [43]. It supports the retrieval of contours and their hierarchical alignment. The modeling use case does not expect users to draw shapes with holes, so that it is not necessary to consider the hierarchy. The function will therefore be called with the parameters `RETR_EXTERNAL`.
In: Binary image
Out: List of contours
8. **Filter Contours:** To reduce the computational load for the following steps of the processing pipeline, the found contours are filtered by its area. All contours with an area smaller than 0.2% of the original source image will be removed from the list of found contours.
In: List of contours
Out: List of contours
9. **Polygon Approximation:** The set of points that defines a contour's outline can be greatly reduced by using the parametric Ramer-Douglas-Peucker [39] algorithm for polygon approximation. The parameter ε , used in the RDP algorithm, defines a threshold used to determine if points of the contour can be discarded. The pipeline uses $\varepsilon = 0.3\% * \text{perim}(c)$,

defining the parameter in relation to the contour's perimeter, which was empirically found to be a good tradeoff between reducing the number of points without losing too much details.

In: List of contours

Out: List of polygons

10. **Filter Polygons:** The final step of the processing pipeline removes all non-simple polygons from the set. A polygon is simple if each vertex is used by exactly two line segments and consists of a single series of connected line segments. Consequently, a polygon may not have holes or have line segments that cross.

In: List of polygons

Out: List of polygons

4.2.3 Modeling User Interface

Once shapes have been captured with the shape capture application, they can be used for modeling by switching to AR modeling mode using the button B_2 , as seen in Figure 4.3. The modeling mode shows immediately the result of the current modeling technique combined with the most recent shape that was captured.

The modeling user interface (see Figure 4.5) is very similar to the interface for capturing shapes. The model d) is displayed in the augmented environment on top of the visual marker a) used for tracking. The axes c), and therefore the parameters that are easily accessible as described in Section 3.3 are displayed at the models position. When a modeling technique's parameter is modified, the related axis is highlighted and the parameter's description is displayed e) on the lower border of the screen. Along the upper border of the device's screen each captured shape is represented by a gray rectangle on a dark bar. The shape currently being used for modeling is highlighted in orange and can be changed by using the arrow buttons alongside the bar. Important functions are accessed by buttons positioned in the screen's corners to maximise the space available for modeling input. The buttons are used to delete the current shape from the set of captured shapes (B_1), switch to capture mode to create a new shape (B_2) and to iterate through the available modeling techniques (B_3). An instance of the model can be instantly created by a single tap on the model.

4.3 Content Distribution

With the modeling functionality new content can be generated and it is desirable to distribute the content in the collaborative workspace amongst its users over the network. So far, neither ARTiFICe, nor Unity itself offer out of the box solutions to distribute arbitrary runtime generated 3D content over the network. However, Unity ships with functionality that can be used to achieve content distribution.

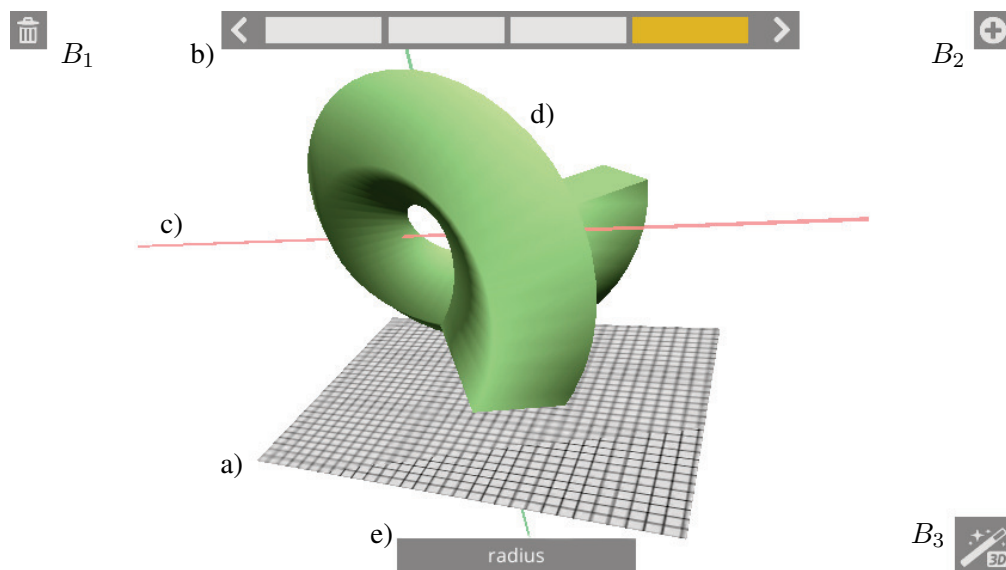


Figure 4.5: Modeling user interface for handheld augmented reality.

4.3.1 Data Transmission Analysis

Non-animated 3D polygonal models basically consist of shape and appearance. The shape is defined by its polygonal mesh. The appearance is influenced by its shape, color or texture and also techniques like e.g. bump or normal mapping. The models that are meant to be distributed are created in the collaborative environment, as described in Section 4.2, and consist only of a mesh and a solid color. The data that needs to be synchronised for a generic 3D model is therefore the mesh consisting of the vertice-, triangle and normal vectors and a color defined by its red, green, blue and alpha (RGBA) components.

Mesh Data Any model's shape can be fully described with the vertice-, triangle and normal vectors of its mesh. Mesh data is widely used in computer graphics applications. However, it can grow large easily even for simple models. Assuming a full 360° ring torus whose outer circle is sampled every 10° and the inner circle every 20° . The number of bytes that needs to be transmitted for the mesh of the given torus is broken down in Table 4.1. In total, the mesh would comprise 38.880 bytes. The mesh's size is driven by the two sampling parameters for the inner and outer circle as they control the granularity of the model and results in a quadratic growth.

Geometric Data To minimise the data that needs to be shared for a model, one could send a geometric build instruction rather than a fully generated 3D model. The ring torus mentioned before can be fully described with a few parameters. The build instruction would in this case consist of six parameters: 1) Geometric-Schematic (i.e. 1:Full Ring Torus), 2) Radius of Inner Circle, 3) Radius of Outer Circle, 4) RGBA Color, 5) Sampling of Inner Circle and 6) Sampling of Outer Circle. Assuming each parameter is encoded as a single float value, the whole model

| Vertice Vector | Triangle Vector | Normal Vector |
|--------------------------|------------------------------|----------------------------|
| 36 * intervals | 36 intervals | 36 intervals |
| 18 vertices/interval | 18 * 2 triangles/interval | 18 * 2 normals/interval |
| 12 byte/vertice | 12 byte/triangle | 12 byte/normal |
| 648 vertices @ 7776 byte | 1296 triangles @ 15.552 byte | 1296 normals @ 15.552 byte |

Table 4.1: Data for a sampled circular thorus mesh.

can be described with only 24 bytes. In this case, by using a geometric description, a reduction to approximately $\frac{1}{1500}th$ of the fully generated 3D model's size can be achieved. The size also remains constant, as a higher granularity does not introduce additional parameters. Knowledge of the model's geometry can therefore tremendously reduce the data that needs to be synchronised in the collaborative environment.

Modeled Data The modeling techniques described in Section 4.2 uses not only a geometric schematic and some parameters, but also a hand-drawn 2D shape to construct the mesh for a model by extrusion. Using a hand-drawn shape gives users more flexibility compared to using a geometric shape. Moreover, the build instruction's size increases, as each point of the 2D hand-drawn shape needs to be saved explicitly. Staying with the circular thorus, the two parameters for the inner circle would be replaced by a set of 18 points assuming the circle is sampled every 20° . The model could then be described with the point set and four parameters: 1) Geometric-Schematic, 2) Radius of Outer Circle, 3) RGBA Color and 4) Sampling of Outer Circle. This sums up to 160 bytes and results in a reduction to approximately $\frac{1}{240}th$ of the fully generated 3D mesh. The size of the model built instruction has linear growth.

Temporal Frequency The distribution of the models occurs in exactly two cases. The most common case is that a new model is shared among already connected users. The model is then sent once to each user. The second case arises, when a new client connects to the collaborative environment. Then, all user generated models are shared with the client and can cause a temporary strong peak in transmission.

Unity allows arbitrary state synchronisation amongst clients and the server in two ways. The following section addresses the general differences between the two and how they can be used to synchronise data of user generated 3D models.

4.3.2 Networking in Unity

For both approaches of networking in Unity, an instance of the type *NetworkView* is used to set up the necessary communication layer between client and server. The first approach uses the paradigm of *Remote Procedure Calls (RPC)*, whilst the other is called *NetworkView Serialisation*. Besides Unity's networking capabilities mentioned before, there exist more flexible solutions for network communication. Unity Pro offers plain TCP or UDP¹ communication us-

¹TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are both end-to-end transport protocols according to the OSI layer. While TCP provides a reliable byte-stream channel and is designed to detect and

ing the *.Net Framework Mono* that is shipped with Unity. Third party addons, like *Photon Unity Networking* offer professional high level solutions. As Unity Pro is not available for free and the free editions of 3rd party addons do not allow deployment to platforms like Android or iOS due to Unity's license agreements, 3rd party addons are therefore not discussed in the following.

Remote Procedure Calls are asynchronous calls to a shared instance of a *NetworkView* between entities (clients and server). RPC calls can be sent from and to any entity that share the *NetworkView*. The key features and requirements of RPC's are:

- Entities must share a *NetworkView*.
- All entities must have the RPC defined on the *NetworkView*.
- The RPC can be called from and to all entities.
- Runtime allows payload size of 4096 bytes for *string*.
- Runtime allows payload size of +10 mbyte for *byte[]*.
- Userdefined datastructures cannot be sent without serialisation.
- RPC's are asynchronous, thus cannot return data.
- RPC's can be chained to return a result.
- RPC's are triggered manually.

NetworkView Serialisation is a one-way communication between the participating entities. The lifecycle of a *NetworkView* begins with its creation on a specific entity which in turn signals the creation to the other entities as well. The entity that initially created the *NetworkView* is its owner and can exclusively send data by using serialisation. All other entities can only receive data. The key features and requirements of NetworkView Serialisation are:

- Entities must share a *NetworkView*.
- The payload that is sent is not limited to a specific size.
- Data can only be sent from the owning entity.
- Entities can be excluded to receive data.
- Data is sent in regular intervals (if changed), the default send-rate is 15Hz.
- The *NetworkView* decides when serialisation has to be performed and cannot be triggered manually.

recover from losses, UDP does not handle transmission errors. TCP is regarded as a connection oriented and reliable transmission channel, whilst UDP is connectionless and unreliable [27].

4.3.3 Distribution Procedure

Both concepts of state synchronisation (RPC, Serialisation) in Unity are one-way communication channels and both concepts could be used in the same way to distribute the model data. However, RPC is in this case the preferable concept. RPC's are asynchronous, can be called easily in both directions and can be chained to return a call's result. Serialisation instead can not be controlled directly and should be used when the synchronised states frequently change. Content distribution using Unity's built-in RPC functionality is implemented in four steps. For an illustration of the involved classes and an exemplary call sequence, see Figure 4.6 and 4.7, respectively.

Involved Classes

- *UGCUploader*: There is exactly one instance running on each client and the server. The instances are connected via a *NetworkView* and the *UGCUploader* is used to initiate distribution from the client to the server with the RPC call *spawn(...)*.
- *UGCHandler*: Instances of this type are created by the server, in case an object containing user generated content shall be created in the environment. The instances are connected via a *NetworkView*. If a client instance has no model data attached, the RPC calls *inquire(...)* and *push(...)* are used to receive the model data from the server.
- *UGCData*: This abstract class is used to serialise and deserialise arbitrary user generated content to the JSON data format.
- *UGCPrimitive*, *UGCModel*, *UGCMesh*: These are specific realisations of *UGCData* that define a user generated model. Each realisation must be able to reconstruct a model of its specific kind with the the *build(...)* procedure. *UGCPrimitive* is a simple geometric descriptor (e.g. Sphere, Cube or Tetraeder), *UGCMesh* can hold arbitrary mesh data and *UGCModel* is a build instruction that uses one of the modeling techniques from Section 4.2.

Procedure

1. *Content Serialisation*: The client, that wants to send a new model to the server, needs to serialise the model data first. Therefore, it has to create an object of the type *UGCData* that will contain all data necessary to rebuild the model. *UGCData* and its realisations are POCOs². Thereby, serialisation and deserialisation can be automatically done using a JSON parser and the C# programming language reflection and introspection capabilities.

²POCO (Plain old CLR Object) is a simple CLR (Common Language Runtime) object that solely depends on the .Net programming framework. Analogously, the term POJO (Plain old Java Object) refers to simple objects used in the Java Runtime Environments. Plain objects are mostly used to provide a simple way for data serialisation and to pass data through application layers [33].

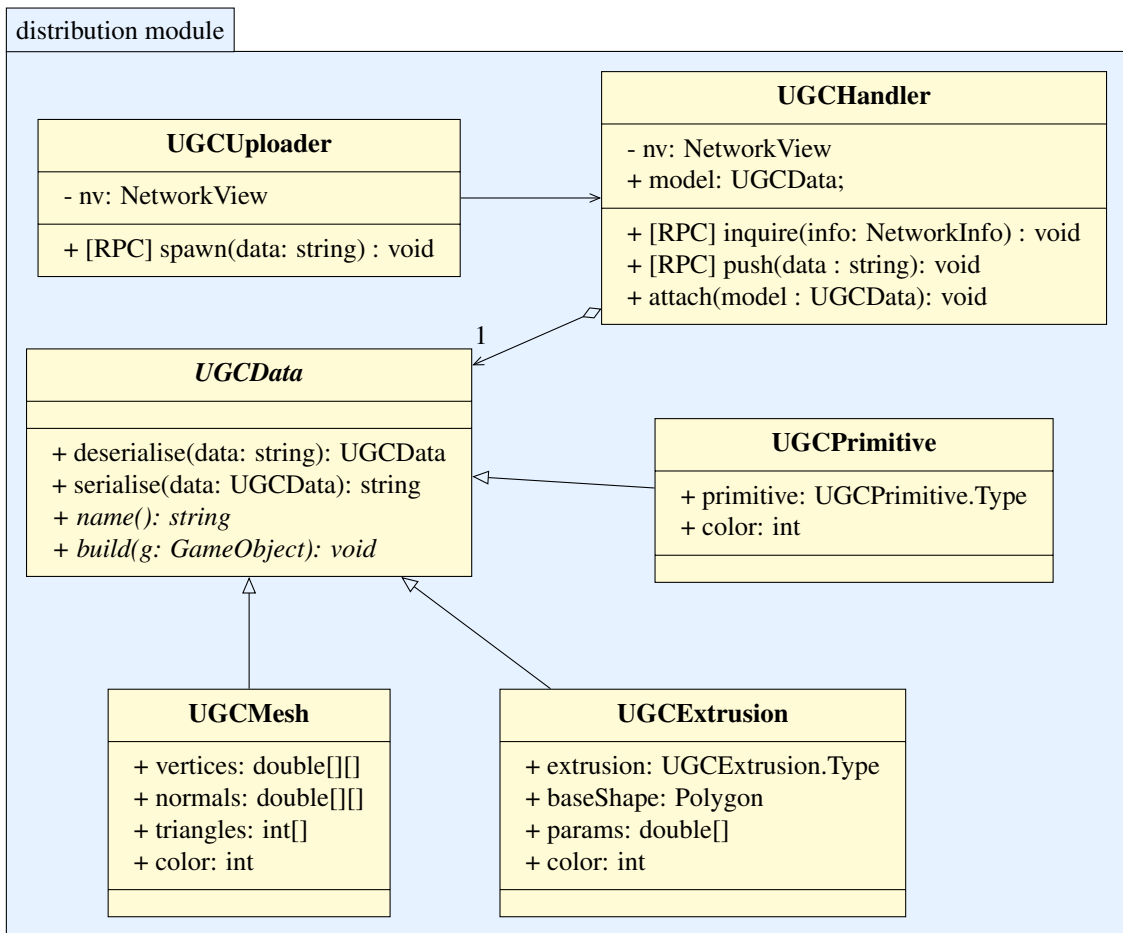


Figure 4.6: Classes involved in the distribution of the user generated content.

2. *Content Upload:* All clients and the server are connected with a *NetworkView* via an instance of *UGCUploader*. When a client creates a new object with a user generated model in the collaborative environment, it will upload the serialised *UGCData* object to the server and ask for its instantiation with the RPC *spawn(...)*.
3. *Network Instantiation:* When the server receives a new upload from a client, it triggers the instantiation of a *UGCHandler* object on itself P_0 and all clients P_i . The object holds a *NetworkView* and has two RPC's registered for client synchronisation. Additionally, the server attaches the previously deserialised *UGCData* object to its local instance P_0 .
4. *Client Synchronisation:* When an object P_i is instantiated on a client, it will inquire the serialised *UGCData* object with a RPC from the server-sided object P_0 which has the model data attached. The server-sided object P_0 , will then perform a chained RPC and

push the data to the client. The client is now synchronised and can build the model from the received data. See Figure 4.7 for the synchronisation call sequence.

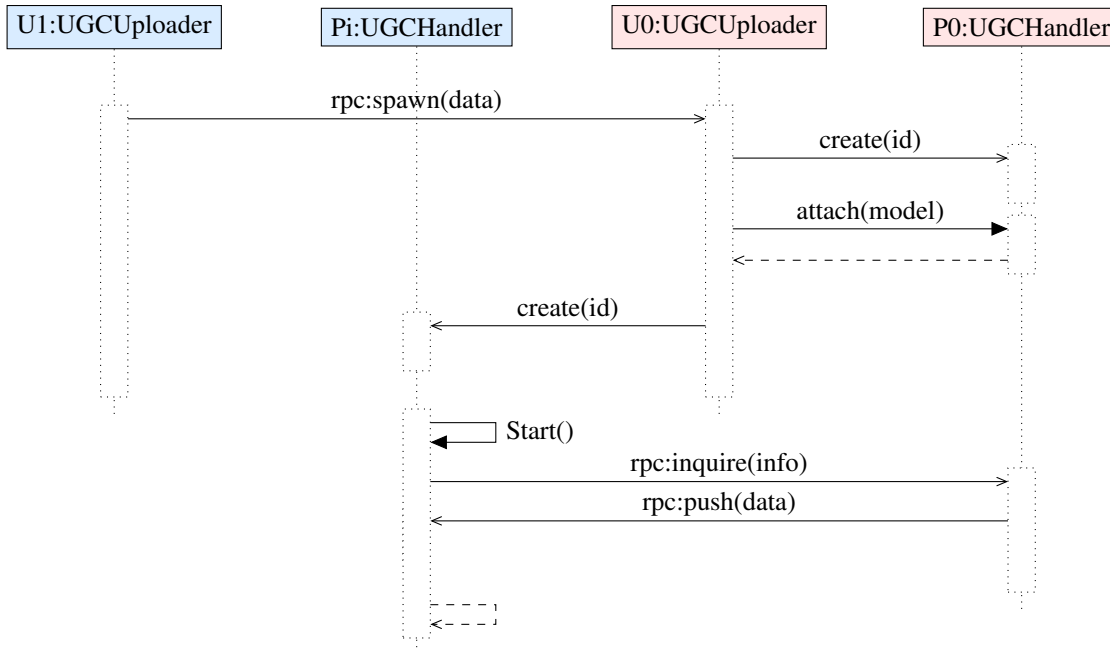


Figure 4.7: Exemplary sequence for the distribution of user generated content. Client-sided classes are colored in blue, server-sided in red. Explicit network communication is prefixed with *rpc*.

4.4 Inter Process Communication

The modeling application and the shape detection application act as a prototype for the *Prefab Based Modeling* technique as described in Section 3.3. As the shapes need to be transferred from the shape detection to the modeling application, it is necessary to use mechanisms for inter-process communication (IPC). The modeling application is developed in C# using the Unity game engine, whilst the shape detection application uses the OpenCV framework and is written in Java. The targeted development platform for both application is the Android operating system and available IPC mechanisms are discussed according to the given situation in this section.

4.4.1 IPC Mechanisms

This section describes the IPC mechanisms that are available on the Android platform.

Android Intents An intent is a simple data object that can be used in Android to contact one application from another. Basically there are three cases in which intents should be used. The most common use is to start another application in the foreground to perform a task and return a result. The second use case is to start a service to perform an operation in the background, e.g. a download, without interrupting the current application. The last case is to send a broadcast message to inform a set of applications about specific events or a state, as e.g. that the system is shutting down [19].

Android Services Android provides services (applications that run in the background) to enable communication between processes or activities. Services can be implemented in different ways [20].

- **Remote Service:** A remote service is an application that runs in the background and provides RPCs that can be accessed by activities of other processes. A service has to define its interface with the Android Interface Definition Language (AIDL) to describe which RPCs are available. Remote services allow intra- and inter-process communication.
- **Bound Service:** Bound services are basically remote services that do not have their interface exposed using the AIDL. Without defining the service's interface using the AIDL, the service can only be used from activities of the same processes. Bound Services provide only a very limited way of IPC, as it only allows intra-process communication.
- **Messenger:** A messenger allows, like remote services, true IPC but without using the AIDL. The Messenger ships with a simple messaging interface instead of exposing an interface that has to be defined by the developer explicitly.

Networking Apart from the IPC mechanisms specific for the Android platform various network protocols like TCP or UDP can be used for IPC as well. Networking allows not only for intra- and inter-process communication, but also for IPC between computers. Many high-level programming languages like Java, C, C++, Objective-C or C# support networking using sockets or a similar concept, thus networking additionally offers a high level of interoperability.

4.4.2 IPC Architecture

At a first glance, the mentioned Android IPC mechanisms do not seem to be applicable, as the modeling and the shape detection application are written in different programming languages. However, Unity provides interfaces [38] to use external Java libraries, thus each mechanism can be used for IPC. On the other hand, the implementation would be bound to devices running Android. Using networking instead would help to build a more platform independent solution. Additionally, testing and debugging handheld applications written in Unity is tiresome as deployment takes up to 90sec and more importantly as it does not support remote debugging of target devices. To overcome this problem, Unity provides an application called *Unity Remote* which is installed on the targeted handheld. An application to be tested is then executed on the development computer but the graphical output is streamed to the handheld and the handheld's

touch input is streamed back to the development computer. Using Unity Remote for testing, the modeling application would no longer be executed on the handheld so that local IPC, and therefore the mentioned Android IPC mechanisms, is not sufficient anymore. Using networking could therefore simplify the development process of the modeling application by eliminating the need to deploy the application to a handheld device for testing purposes. The application can be tested right on the development computer with *Unity Remote*.

Therefore, the modeling prototype uses networking as an IPC mechanism to provide a higher level of abstraction and to simplify the development process. The two applications communicate through a communication agent as they are not executed concurrently. The communication agent provides an interface to store, delete and retrieve shapes that are used for modeling and acts as a *storage service*. Communication is performed using a client/server setup and TCP sockets. A special constraint of using existing frameworks imposes Unity, as it provides only a limited Mono 2.0 subset that can be used for the development of Android platforms. To the knowledge of the author, no compatible cross-language communication framework exists and pure socket communication is tiresome. Therefore, the *JSON RMI* framework was developed and is presented in the following section.

4.4.2.1 JSON Remote Method Invocation

JSON RMI is a framework that enables communication between two processes via TCP. The framework is designed to cause minimal programming effort. A remote interface is defined as easy as defining a class interface in C# or Java and does not need an additional interface definition language like AIDL (Android) or IDL (CORBA). For data transmission, the framework uses, as its name already suggests, the JavaScript Object Notation (JSON). The core elements of the framework are the two classes *SimpleRMIService* and *SimpleRMIClient* that handle the connection establishment and termination over TCP, the message interpretation and the procedure call delegation using the reflection capabilities. The framework is developed both in Java and C#.

Class Overview

- **SimpleRMIClient:** Provides client functionality to use a RMI service.
- **SimpleRMIService:** Provides server functionality to create a RMI service.
- **JSONRequest:** An internal data structure for communication from client to server. The structure encapsulates the name of the procedure, its input arguments and configuration parameters of a RMI.
- **JSONResponse:** An internal datastructure for communication from server to client to return the result of the preceding RMI. The structure encapsulates the result of the call or in case of an error an error message. A response is only returned if the call was configured to be a synchronous call.

- **JSONUtil:** A utility class to handle the (de)serialisation of the exchanged data. The class uses the LitJSON [3] library in C# and for Java the Jackson [11] Java JSON-processor.
- **JSONException:** This exception is raised if exceptions occur during transmission or (de)serialisation of the exchanged data.

In the following sections, the implementation of the communication agent used for inter-process communication between the *Modeling App* and the *Shape Detection App* is described.

4.4.2.2 Storage Service

Using the JSON RMI framework, a *storage service* is implemented. The functions and data structures of the service are displayed in Listing 4.1 and 4.2. The service provides four functions to store and access *polygons*. A polygon is here defined by a unique identifier, a set of coordinates $\in \mathbb{R}^2$ as well as the width and height of the source image in which it was detected.

```

1 package de.venditti.shaperecognizer.rpc.shared;
2 public interface IShapeStorage {
3     public void addPolygon(Polygon p);
4     public void deletePolygon(String guid);
5     public Polygon[] getPolygons();
6     public void clearPolygons();
7 }

```

Listing 4.1: Interface of the storage service.

```

1 package de.venditti.shaperecognizer.rpc.shared;
2 public class Polygon {
3     public String GUID = "";
4     public int srcWidth = 0;
5     public int srcHeight = 0;
6     public double[] x = {};
7     public double[] y = {};
8 }

```

Listing 4.2: The data structure used by the storage service.

To create a service that fulfills the interface of Listing 4.1 and is accessible over TCP on port 4321 only a few calls are required (see Listing 4.3). In the lines 12-17, a new *SimpleRMIService* is created. It expects an arbitrary name, the interface to be published and an object implementing the logic of the storage service. The actual implementation of the storage service is shown exemplary in lines 5 and 14-29. The *SimpleRMIService*-object will create a server socket listening on the specified port and delegate valid RMI's to the storage service implementation by using reflection.

```

1 import de.venditti.jsonrpc.SimpleRMIService;
2 import de.venditti.shaperecognizer.rpc.shared.IShapeStorage;
3 import de.venditti.shaperecognizer.rpc.shared.Polygon;
4
5 public class StorageService implements IShapeStorage {
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

8   static SimpleRMIService myService;
10
12  public static void main(String[] args) {
14      StorageService serviceLogic = new StorageService();
16      myService = new SimpleRMIService(
18          "Speicher Dienst",    // a arbitrary name for this service
20          IShapeStorage.class, // interface to be published
22          4321,                // listening port
24          serviceLogic         // object implementing the interface
26      );
28
30      myService.start();        // start the service
32      while(myService.isRunning()) Thread.sleep(250);
    }

    // implementation of the storage service's logic
    @Override
    public void addPolygon(Polygon p) { /*...*/ }
    @Override
    public void deletePolygon(String guid) { /*...*/ }
    @Override
    public Polygon[] getPolygons() { /*...*/ return null; }
    @Override
    public void clearPolygons() { /*...*/ }
}

```

Listing 4.3: A storage service with JSON RMI.

4.4.2.3 Storage Client

The implementation of a client that communicates with the storage service is similar to creating the service. Implementations in C# and Java are given as follows. The storage client is created by extending the *SimpleRMIClient* class that is used to establish the connection and to trigger the remote method. The endpoint to communicate with the storage service is defined by providing the port, IP address as well as the service interface of the storage service in the constructor of the base class. The base class also provides *makeRMI* helper functions to form and send valid RMI-requests to the service. A developer only has to pass the input parameters that are meant to be transmitted and an integer specifying the timeout in milliseconds the client shall wait for a response from the server (e.g. line 11 in Listing 4.4). Asynchronous calls can be placed by using a timeout smaller or equal to 0. When called, the helper function inspects the method from which it is called using reflection to place the corresponding RMI. The listings 4.4 and 4.5 show the client implementations for the storage service in Java by using inheritance and in C# with delegates.

```

1  import de.venditti.jsonrpc.SimpleRMIClient;
2  import de.venditti.shaperecognizer.rpc.shared.IShapeStorage;
3  import de.venditti.shaperecognizer.rpc.shared.Polygon;
4

```

```

public class ShapeStorageJSONClient extends SimpleRMIClient implements
    IShapeStorage {
6   public ShapeStorageJSONClient() {
        super(4321, "127.0.0.1", IShapeStorage.class);
8       Connect();
    }
10  @Override public void addPolygon(Polygon p) {
        makeRMI(1000, p);
12  }
14  @Override public void clearPolygons() {
        makeRMI(1000);
    }
16  @Override public Polygon[] getPolygons() {
        return makeRMI(1000, Polygon[].class);
18  }
}

```

Listing 4.4: Storage client in Java using inheritance.

```

1  using System;
   using de.venditti.shaperecognizer.rpc.shared;
3  using de.venditti.jsonrpc.general;

5  public class JsonShapeStorageClient : IShapeStorage{
        SimpleRMIClient<IShapeStorage> myClient = new SimpleRMIClient<IShapeStorage
            >(4321, "127.0.0.1");
7   public JsonShapeStorageClient() {
        myClient.Connect();
9   }
11  public void addPolygon(Polygon p) {
        myClient.makeRMI(1000, p);
    }
13  public Polygon[] getPolygons() {
        return myClient.makeRMI<Polygon[]>(1000);
15  }
17  public void clearPolygons() {
        myClient.makeRMI(1000);
19  }
}

```

Listing 4.5: Storage client in C# using delegation.

4.5 Selection & Manipulation

The selection and manipulation techniques that are part of the user study are integrated into the Artifice framework. So far, Artifice did not differ between selection and manipulation from an implementational point of view. Artifice provides a single base class handling logic common to all interaction techniques and new techniques are created using inheritance. For the integration of the three selection techniques (RayCast, Expand, DrillSample) and the two manipulation techniques (MultiTouch3D, HomerS), the composite design pattern [13] seems more appropriate since combinations of different selection and manipulation techniques are desirable.

4.5.1 Composite Interaction

Using the concept of composition, makes it not only easier to use and combine selection and manipulation techniques once they are implemented, but also simplifies future implementations. Composition enforces the separation of responsibilities and therefore reduces the size of logical code blocks, such as functions or classes, thus makes the source code easier to maintain.

To separate the responsibilities of a selection and manipulation technique, it is important to analyse the states each component passes and how they interact with each other. From the perspective of a sole selection technique there are four states: 1) Nothing Selected, 2) Refining, 3) Selected and 4) Deselected. For a manipulation there are just two states: 1) Choose Manipulation and 2) Manipulating. The list of states proposed for the composite interaction and valid transition within these states are depicted in Figure 4.8. The transitions in the state diagram reflect the interplay between a selection and a manipulation technique and is the logic that is implemented in the composite interaction. The states and the transitions are explained in the following enumeration. The selection technique is responsible for all transitions that are drawn with a continuous line, while the dashed transitions are under responsibility of the manipulation technique.

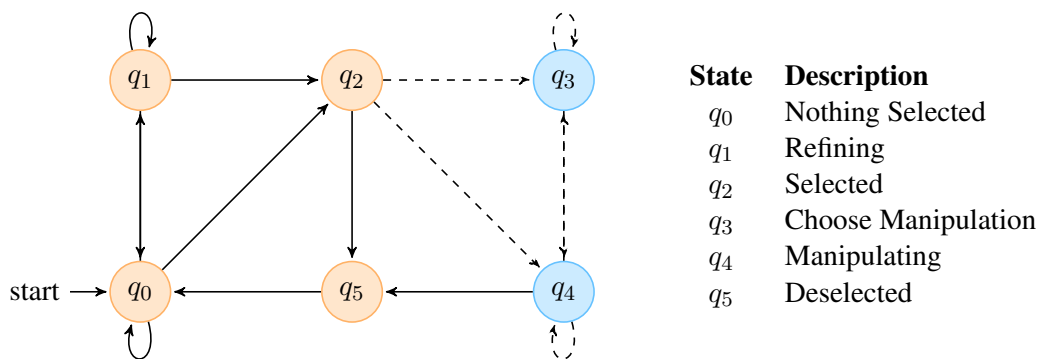


Figure 4.8: Interaction State Diagram.

q_0 **Nothing selected:** It is assumed that nothing is selected at the beginning. Once a selection is triggered, there are three options. First, the state remains in q_0 if the selection was not successful. Second, if multiple objects were selected, it switches to q_1 to perform a refinement of the selection, or third, it switches to q_2 if a single object was selected.

q_1 **Refining:** In the refinement mode, a user may either cancel the selection process and return back to q_0 , inspect the multiple selected objects and remain in q_1 , or select one of the objects presented in the refinement and proceed to q_2 .

q_2 **Selected:** Once an object is selected, a manipulation mode may be chosen by switching to q_3 or immediately start manipulation in state q_4 with a predefined manipulation mode. Alternatively, the selected object may be deselected as well and switch to q_5 .

q_3 **Choose Manipulation:** The system may remain in state q_3 until a manipulation mode is chosen, at which it switches to state q_4 .

q_4 **Manipulating:** During manipulation users may change the manipulation mode and return back to q_3 , perform manipulations and stay in q_4 or finish the manipulation by deselecting the object and switch to state q_5 .

q_5 **Deselected:** Once an object has been deselected, the system returns back to the initial state q_0 , where nothing is selected.

To implement an interaction technique with exchangeable selection and manipulation capabilities, a generic class *CompositeInteraction* is derived from Artifice's *InteractionBase*. This ensures compatibility with existing interfaces and design decisions. However, the *CompositeInteraction* holds no logic regarding a specific selection or manipulation technique, but only logic to redirect calls to its components (as seen in Figure 4.8), which in turn hold the specific implementation of a selection- and manipulation technique. To redirect the calls it is necessary to define interfaces between the component and the compositor. The two interfaces that are used by the compositor are *SelectionTechnique* and *ManipulationTechnique*.

4.5.2 Selection Interface

A selection technique is defined with the interface *SelectionTechnique* and characterized by the following three methods.

- *public void ShowGUI(InteractionState s)*: This procedure is called from the compositor's graphical user interface update cycle. It should be used to display information and interaction controls (windows, menus, buttons) on the 2D overlay of the AR view. The procedure's only argument is the current state of the composite interaction so that the realising technique can adjust its GUI accordingly. It should be noted that this function might be called multiple times per frame.
- *public InteractionState UpdateSelect(InteractionState s)*: This function is called from the compositor's rendering update cycle exactly once per frame. It should be used to perform the actual selection, refinement or deselection of the desired object. The function is passed the current interaction's state and in turn returns the future state according to its logic and the user's interactions. If the given state is not handled or the technique is waiting for user interaction to perform a state change, the function has to return the unchanged state.
- *public GameObject SelectedObject()*: Is used to retrieve the selected object upon a successful selection. A selection technique must always hold the currently selected object or *null* if nothing is selected.

The methods *ShowGUI* and *UpdateSelect* are called regularly from the Unity Game Engine's update cycles regardless of the current state of the composite interaction. Therefore, both methods must always evaluate the current state and act accordingly.

4.5.3 Manipulation Interface

Similar to selection techniques, the interface for manipulation techniques features functions that are called from the compositor's GUI and rendering update cycles.

- *public void ShowGUI(InteractionState s, GameObject g)*: Is called in exactly the same context as the function *ShowGUI* for selection techniques. The currently selected object is passed as an additional parameter. This might be used to show specific information about the selected object. Currently, it is used to display GUI controls, to choose a manipulation type.
- *public InteractionState UpdateManipulate(InteractionState s, GameObject g)*: This function is used to implement the specific manipulation based on the user's interactions with the device. It is not meant to directly modify the object, instead the manipulation should be stored in a local variable for later retrieval.
- *public Manipulation GetManipulation()*: The manipulation that shall be applied to the object has to be returned by this function. A *Manipulation* is defined by a translation and local scale vector as well as a rotation angle and axis. The values are understood to be relative to the objects pose upon selection.
- *public ManipulationType GetManipulationType()*: This function returns the type of the manipulation that is meant to be applied to the object. The following manipulation types are defined: translate, rotate, translate & rotate, scale.
- *public bool supports(ManipulationType t)*: Every *ManipulationTechnique* must return if it supports the manipulation of the given type. This simplifies the development of a generic user interface to choose the manipulation type.
- *public void disable(ManipulationType t, bool b)*: In some scenarios it might not be desired to allow all types of manipulation. This function is used to disable certain manipulations at runtime to limit the number of available choices upon selection.

The methods *disable*, *GetManipulation* and *GetManipulationType* are independent to the concrete realisations of a manipulation technique, thus they can be shared among all implementations. Therefore, a manipulation technique is not defined with an interface but as an abstract class. Only the methods *UpdateManipulate* and *supports* are abstract and have to be implemented by all realisations. Optionally, the *ShowGUI* procedure can be overridden to provide a different user interface. An overview of the classes involved in the composite selection and manipulation concept can be found in Figure 4.9.

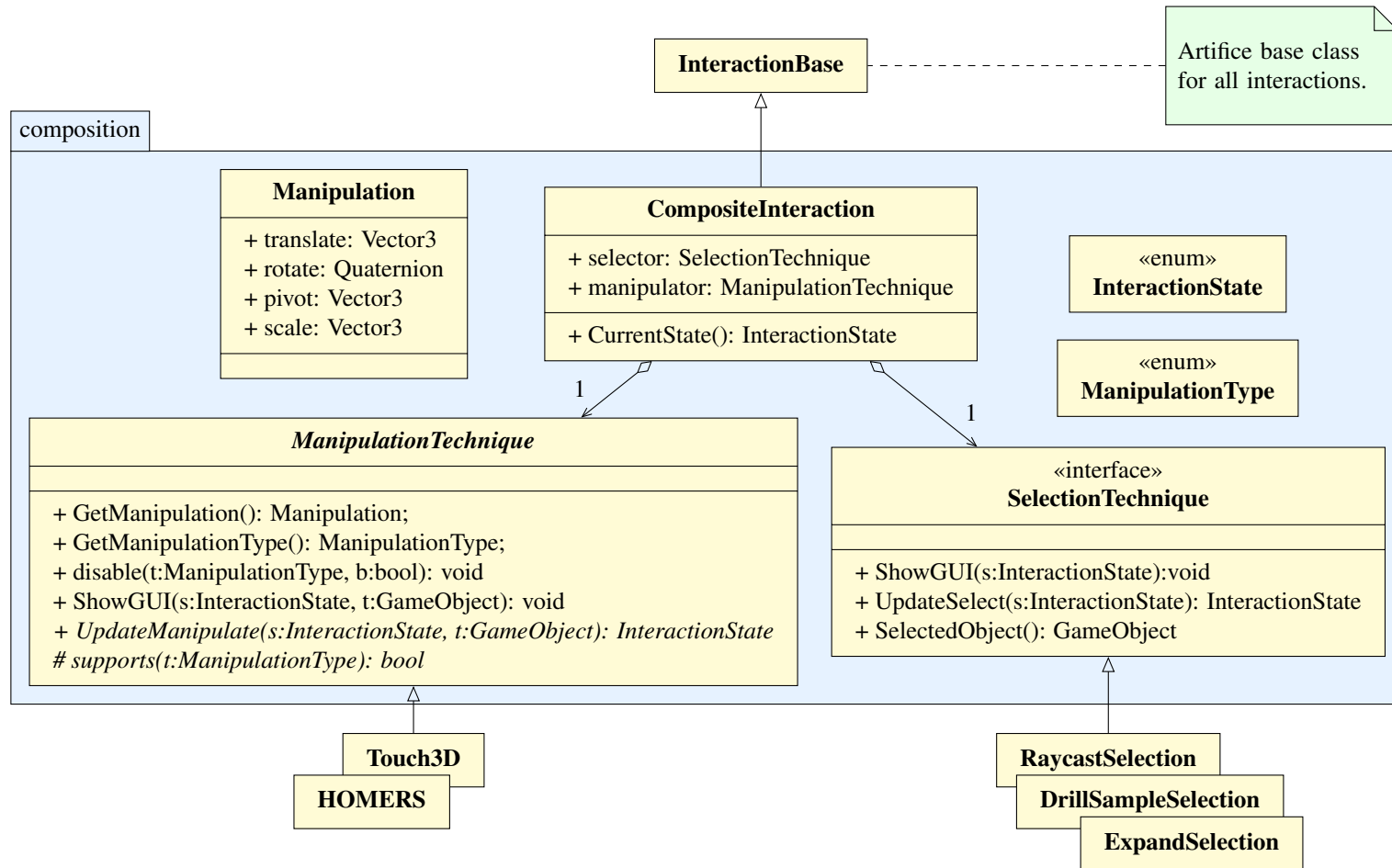


Figure 4.9: Class diagram for composite interaction techniques in Artifice.

4.6 UserStudy Application

To conduct the user study for the selection and manipulation techniques proposed in Sections 3.4 and 3.5, an application is implemented. The application's purpose is to guide users through the test scenarios defined in Section 5.4.2 and 5.5.2, to randomise the test sequences and to record and save data for the quantitative evaluation. At the beginning of each scenario a simple description of the upcoming task is displayed. The participants may inspect, without being able to interact, the virtual environment in order to understand the task. Once they feel to understand the task, they can proceed to start the actual test.

Workflow Once the test starts, the application generates a new test record and a random unique identifier which is used to link the collected test data to the answers from the questionnaires the users are asked to complete. The entire questionnaire is attached in the Appendix under Figures 7.1-7.12. Interrupted test may also be resumed so that already completed scenarios do not have to be repeated. The test is structured in as many sessions as there are interactions. The manipulation test has two sessions with four tasks and the selection test has three sessions with three tasks. The workflow is therefore designed to first, randomise the order of the different techniques and secondly the tested scenarios. This allows users to consecutively complete all scenarios for a given technique. The workflow of the application is illustrated in the flowchart in Figure 4.10.

Data Acquisition The application makes performance measurements and saves the collected data for the statistical evaluation. For each successfully finished test scenario a record is saved on the handheld device, as shown in Listing 4.6. The record contains administrative information, such as the creation time (2), the user's unique identifier (3), the name of the tested technique (4) and the tested task (5). The performative measures that are collected are the duration upon completion (6), the number of selections (8), which and for how long a manipulation (9ff) was used. As users are able to pause the test, each measured duration is saved with and without the corresponding correction, as e.g. in lines 10 and 11.

```
1 // filename: MAN_9R9HS6_HOMERS_ManipulationTask3.json
  "time": "5/5/2014 12:04:54 PM",
3  "uid": "9R9HS6",
  "technique": "HOMERS",
5  "test": "ManipulationScenario3",
  "duration": 26540,
7  "durationWithoutPause": 26520,
  "selections": 1,
9  "interactions": [{
    "duration": 22400,
11   "durationWithoutPause": 22370,
    "type": "TRANSLATEROTATE"}]
```

Listing 4.6: Recorded test result for a single test scenario.

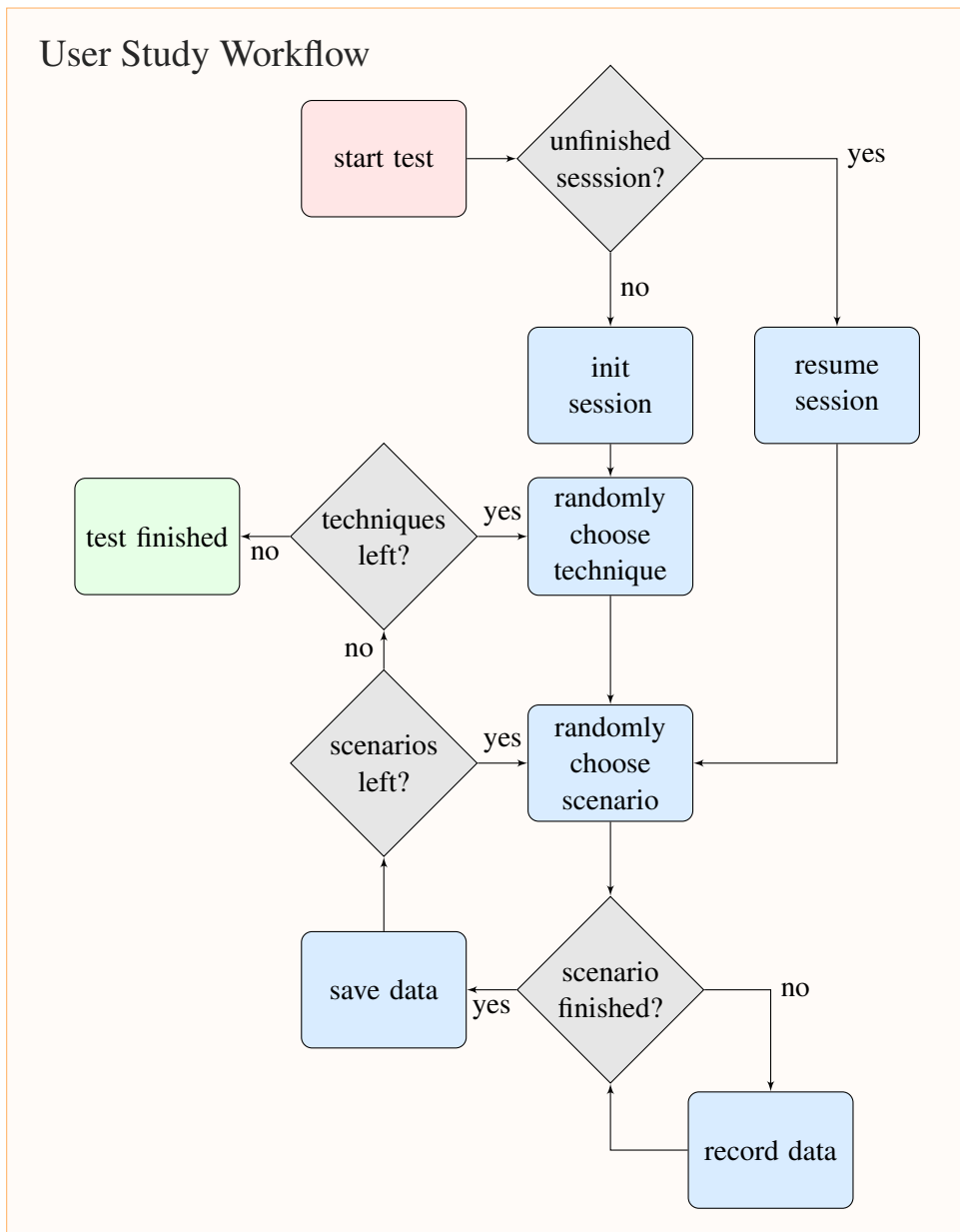


Figure 4.10: General workflow of the application used in the User Study to collect the relevant quantitative data and to guide the users through the study.

User Interface The application’s user interface is designed to guide the participants through the test procedure and to inform them about the current state of the test. Using an automatic test procedure helps to ensure equal test conditions for all participants, thus makes the test results more reliable. The five distinctive screens that guide the participants are presented in Figures 4.11ff.

This is the first screen of the user study. Users are informed about the general test procedure that once confirmed, proceeds to the first task.

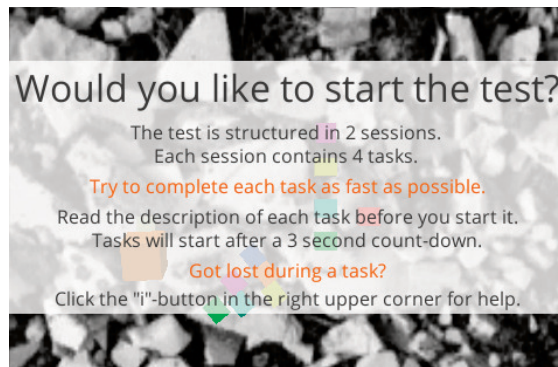


Figure 4.11: Introductory screen of the user study.

Users may resume a previously interrupted test. The application asks only once to resume the unfinished session. If not resumed, the incompleted test results are discarded.

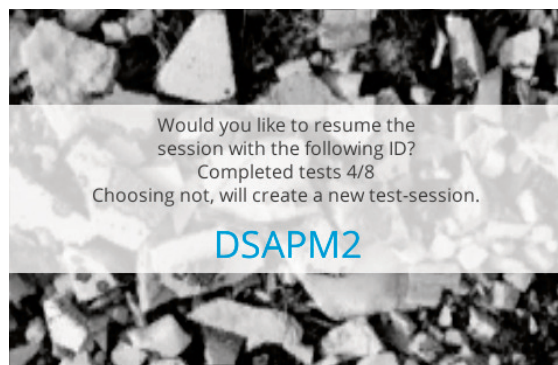


Figure 4.12: Resume an interrupted test.

Each upcoming task is introduced by a short explanatory text. The information states which interaction is to be used and details about its solution. The task starts after a three second countdown upon confirmation.

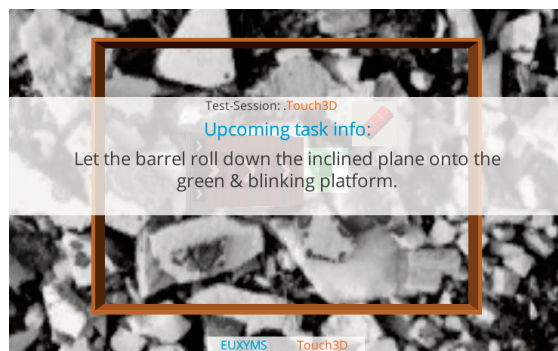


Figure 4.13: Explanatory text for a specific task.

While solving a task, users may pause the test at any time to read the preliminary information of the current task. However, the AR environment is hidden.

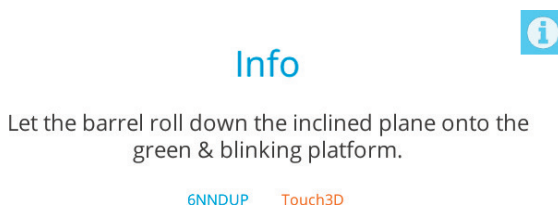


Figure 4.14: Pausing the test.

When all tasks have been completed, an informational screen is displayed. The unique identifier referencing the results of this test is displayed and users are asked to write the ID into their questionnaire.

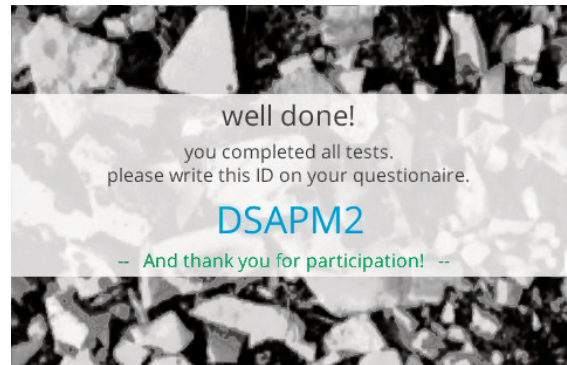


Figure 4.15: End of the test.

Evaluation & Discussion

This chapter presents quantitative and qualitative measures for the shape detection pipeline, exemplary 3D models as well as the results of the thorough statistical evaluation of the user study for the selection and manipulation techniques.

5.1 Shape Detection

The performance of the processing pipeline used to detect hand-drawn shapes is highly dependent on the parameters of its processing steps and the lighting condition in which it is used. The most important parameters are therefore quantitatively and qualitatively evaluated under different conditions. The values used as fixed thresholds for the Canny edge detection in the following evaluation are chosen empirically and were found to provide satisfying results. Furthermore, the employed test pattern next to Table 5.1 is designed by the author to perform reproducible test measures.

5.1.1 Per Pixel Operations

The operations 1) Preprocessing, 2) Subsampling, 3) Noise Reduction, 4) Threshold Estimation, 5) Canny Edge Detection, 6) Closing Operation and 7) Find Contours are all per-pixel operations. As such, their execution time depends only on the size of the image, except for 7) which also depends on the content processed. Image subsampling has a significant impact to the pipeline's throughput but also affects the quality of the detected shapes. Therefore, the average execution time per processing operation is measured with (t_b) and without (t_a) subsampling. Table 5.1 shows the results of the measurements. The camera's source image had 864x480 pixels and the pipeline used fixed Canny thresholds with $t_1 = 10$ and $t_2 = 50$. All values are stated in [$ms/frame$]. Nine black circles with a diameter of 1.2cm and a line width of 2pt were used as a test pattern (see Table 5.1).

The image applied subsampling reduces the initial source image to a quarter of its original size. Consequentially, the execution time for the operations 1-6) drop roughly to a quarter com-

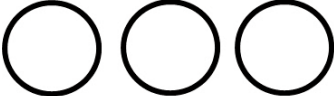
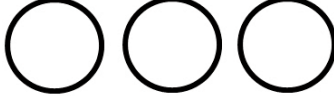
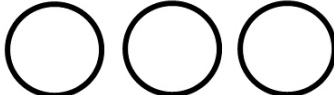
| Task | t_a | t_b | |
|--------------------------|-------|-------|---|
| 1) Preprocessing | 3.6 | 3.6 |  |
| 2) Subsampling | - | 2.0 | |
| 3) Noise Reduction | 16.4 | 4.0 | |
| 4) Canny Edge Detection | 47.6 | 9.3 |  |
| 5) Closing Operation | 33.3 | 8.1 | |
| 7) Find Contours | 7.8 | 3.4 |  |
| 8) Filter Contours | 0.5 | 0.7 | |
| 9) Polygon Approximation | 0.7 | 0.9 | |
| 10) Filter Polygons | 2.5 | 3.0 | |
| Σ | 113.0 | 34.9 | |

Table 5.1: Average execution time [ms] per operation and frame with and without subsampling.

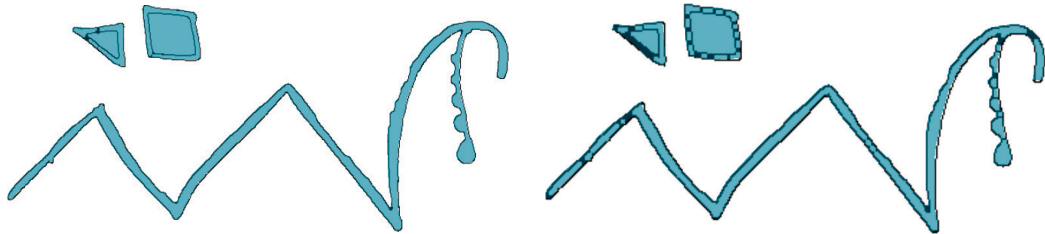


Figure 5.1: Result of Canny edge detection with having subsampling disabled (left) and enabled (right). Quality loss caused by reducing the image size.

pared to the execution time t_a without subsampling. Subsampling the image allows in this case to perform the shape detection with responsive $29fps$ instead of $9fps$ and at the same time use a high quality video feed for the AR interface. However, using a smaller image for the shape detection causes the resulting shapes to have fewer details (see Figure 5.1).

5.1.2 Per Point Operations

The operations 7) Find Contours, 8) Filter Contours, 9) Approximate Polygons and 10) Filter Polygons are operations that are executed on a set of (possibly) connected points. The average execution time of these steps depend therefore on the content of the image and the shapes it shows. Figure 5.2 depicts measurements for the execution time per frame of the per-point operations 7-10 depending on the number of shapes visible in the camera image. The shapes used in the test pattern were again black circles with a diameter of $1.2cm$ and a line width of $2pt$. The pipeline was fed with frames at a resolution of 864×480 pixels and had fixed Canny thresholds with $t_1 = 10$ and $t_2 = 50$ and subsampling enabled.

Each circle was detected with approximately 25 points. According to Figure 5.2, all operations, except for 7), seem to have linear growth with Find Contours at $0.26ms$, Filter Contours at $0.05ms$, Approximate Polygons at $0.05ms$ and Filter Polygons at $0.2ms$ per shape. The operation 7) that finds the contours, has linear to asymptotic growth, as it is not purely an operation

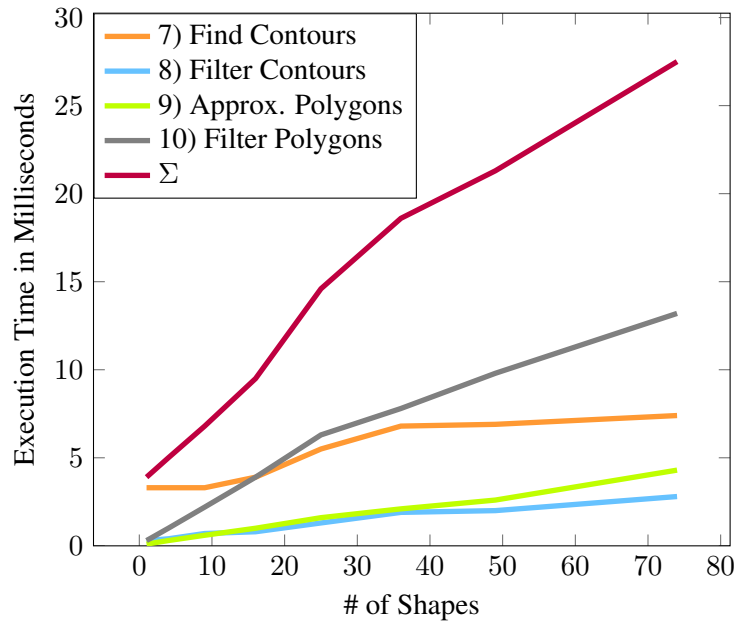


Figure 5.2: Execution Time for Per Point Operations.

that is executed per point but also per pixel. The worst case execution time for operation 7) is therefore defined by the image size it is executed on.

5.1.3 Detection Robustness

The core of the detection pipeline are operations 5) Canny Edge Detection and 7) Find Contours. The OpenCV function *Imgproc.findContours(...)* is used to retrieve the actual contours from the camera image in 7), using a border following algorithm of Suzuki [43]. However, its parameters do not affect the robustness of the shape detection. Thus, the robustness of the shape detection is driven by the quality of the result of the Canny edge detector in 5). Edge detection can be difficult in low contrast scenarios in which a shape's contours are hard to differ from the background. To improve detection robustness the pipeline uses 3) Noise Reduction, 4) Threshold Estimation and a 6) Closing Operation. The impact of each processing step is evaluated as follows.

To test the robustness of the pipeline, the detection rate is measured for different contrast scenarios. Figure 5.3a shows the test pattern that is used to perform the test. The pattern holds a 4x4 grid of circles with 1.2cm in diameter. The circles' brightness decreases from top to bottom and the line width decreases from left to right.

Adaptive Hysteresis Thresholds A major difficulty of using the Canny edge detector is that it is difficult to find thresholds that work well to extract connected structures from an unknown image. The thresholds depend much on the image's contrast, so that using constant thresholds will most likely provide unsatisfactory results. The thresholds $t_1 = 10$ and $t_2 = 50$ were found to be acceptable for conditions with natural daylight as well as light from a traditional 60W

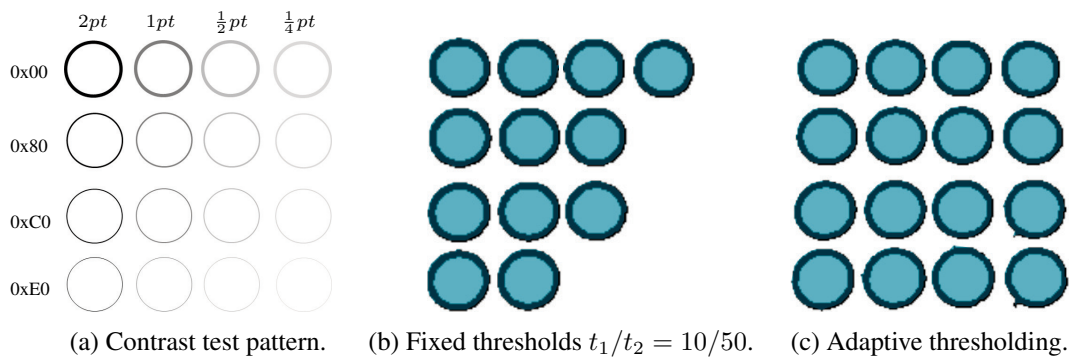


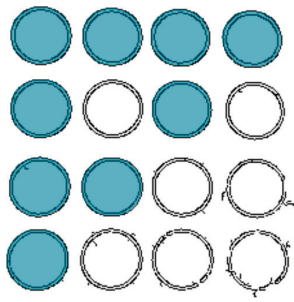
Figure 5.3: Detection performance for fixed and adaptive hysteresis thresholds.

filament lamp, given that the shapes were drawn on plain white paper using a black pen with a $1pt$ wide stroke. To improve the extraction of the Canny edge detector in adverse conditions, the thresholds are estimated for each frame, as described in Section 4.2.2. Figures 5.3b and 5.3c show the result of the detection performance using the stated fixed thresholds (left) and adaptive thresholds (right). The thick black strokes are the result of the Canny edge detector and the blue overlay indicates the detected shapes of operation 7). As can be seen, the fixed thresholds fail to detect the circles with a low contrast and the estimated hysteresis thresholds outperform the fixed thresholds with 16/16 versus 12/16 detected circles.

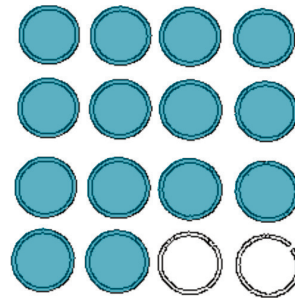
Noise Reduction and Closing Operation More important than the detection of shapes under adverse conditions is to robustly detect and redetect shapes to provide a steady and robust experience for the user. Two consecutive frames delivered by the camera are never the same as they are subject to changes introduced by varying light conditions and noise of the camera’s image sensor. The effect of these irregularities cause the edge detector to falsely classify some of the image pixels. To improve the robustness of the pipeline, the irregularities are reduced using a normalising 3×3 box filter before edge detection and a morphological closing operation after the edge detection. The influence of the irregularities as well as the means to reduce them are tested with the low contrast pattern in Figure 5.3a.

False negative classifications cause structures to have holes and false positive causes structures to “bleed out” (see Figure 5.4a). The less the contrast of a shape is the more it suffers from the randomly occurring irregularities and causes the shape to be detected only sporadically. Using a normalising box filter helps to remove false positive detections and results in a much cleaner edge detection (see Figure 5.4b). Furthermore, using a closing operation after the edge detection helps to refill holes in the structures (see Figure 5.4c), yet the structures have deformations. By applying both operations it is possible to limit the effect of false positives and false negatives to provide a robust experience (compare Figures 5.4a and 5.4d with 9/16 versus 16/16 detected circles).

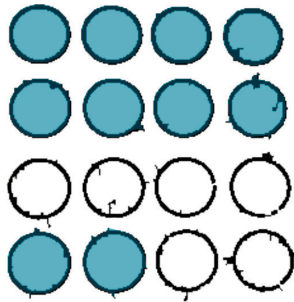
Both operations help to increase the robustness of the shape detection, however, they also cause the detected shapes to have softened corners, as depicted in Figures 5.5a and 5.5b. The left image uses neither noise reduction, nor the morphological closing operation. Here, the shape of the zigzag line is outlined sharper, compared to the right image where both operations are applied.



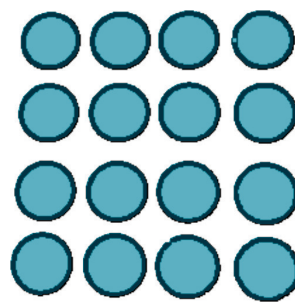
(a) Noise Red.: no, Closing: no



(b) Noise Red.: yes, Closing: no



(c) Noise Red.: no, Closing: yes



(d) Noise Red.: yes, Closing: yes

Figure 5.4: Results of Noise Reduction and Morphological Closing Operation.



(a) Noise Red.: no, Closing: no



(b) Noise Red.: yes, Closing: yes

Figure 5.5: Drawback of Noise Reduction, Morphological Closing Operation.

5.2 Modeling

To outline the capabilities of the *Prefab Based Modeling* technique that are presented in Sections 4.2.3 and 3.3, a few example models are shown in Figures 5.6ff. All models are created in less than a minute on a Samsung Galaxy S II I9100, as described in Section 4.1.1.

Shape: 33 points
 Mesh: 1486 verts, 2970 tris

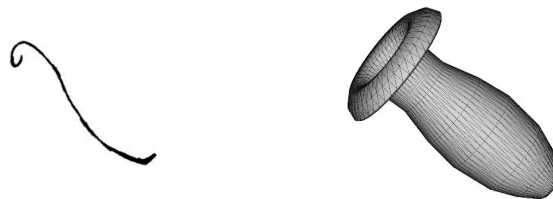


Figure 5.6: 360° Lathe, Vase.

Shape: 23+16 points
Mesh: 92+64 verts, 88+60 tris

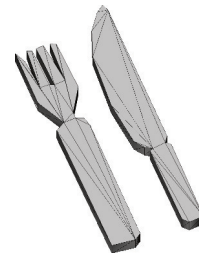
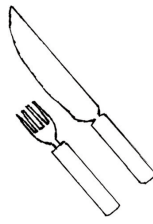


Figure 5.7: Extrusion, Fork & Knife.

Shape: 189 points
Mesh: 765 verts, 752 tris

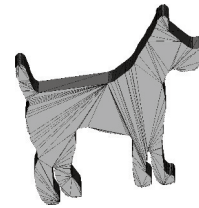


Figure 5.8: Extrusion, Dog.

Shape: 33 points
Mesh: 4555 verts, 8972 tris

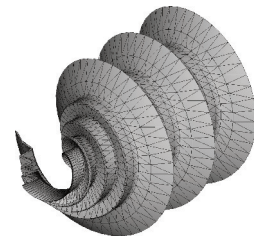


Figure 5.9: 1080° Lathe, Spiral.

Shape: 10 points
Mesh: 381 verts, 716 tris

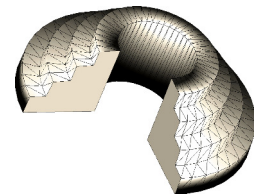
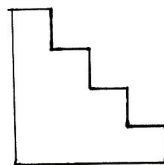


Figure 5.10: 270° Lathe, Stairs.

Shape: 62 points
Mesh: 248 verts, 244 tris

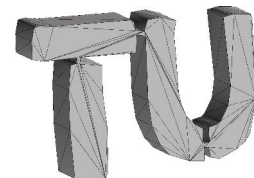


Figure 5.11: Extrusion, TU Logo.

5.3 Experimental User Study

The proposed selection technique *DrillSample* as well as the manipulation techniques *3DTouch* and *HOMER-S* are evaluated in two separate experimental user studies to investigate their performance and user acceptance. The studies use a within-subjects factorial design at which the independent variables are the interaction techniques and the different tasks. The dependent variables are number of interaction steps and task completion time. Additionally, user preferences are measured regarding speed, accuracy and usability.

5.3.1 Design & Procedure

Both studies are structured in five phases (see Figure 5.12) and the participants needed approximately 25 minutes for each study. In the first phase of the study, users are asked to read and sign a standard consent prior to their participation followed by a pre-questionnaire (see Table 5.2) asking questions (see Figures 7.1, 7.2, 7.7 and 7.8) about age, gender and prior experience with smartphones and handheld 3D gaming. In the second phase, the participants get a detailed description of the practical part of the experiment regarding “Selection/Manipulation in Handheld AR” followed by a coached introduction how to use the device and the involved techniques. The third phase consisted of a five minute practice to familiarise with the setup and then different techniques. In the fourth phase, the practical part of the experiment takes place in which participants are neither interrupted, nor given help for the time of the experiment. The practical experiment raises quantitative data, such as completion time or the number of attempts it takes the user to complete the specific challenge. The last phase uses a post-questionnaire (see Tables 5.5 and 5.8) to raise subjective information, like most desired technique and ratings of usefulness as these cannot be retrieved from performative measures.

- Q_1 What is your gender?
- Q_2 How old are you?
- Q_3 About how often do you play video games?
- Q_4 What percentage of your gaming is playing handheld 3D games?
- Q_5 Do you have a multi-touch Smartphone?
- Q_6 Do you have any flexibility or pain issues with your primary hand, fingers or arm?

Table 5.2: Selection and manipulation technique pre-questionnaire

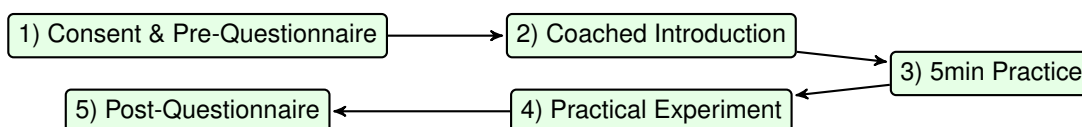


Figure 5.12: User study procedure.

| | | inexperienced | experienced |
|----|--------------------|---------------|-------------|
| a) | Handheld 3D Gaming | 12 | 16 |
| b) | Smartphone | 7 | 21 |

Table 5.3: Users grouped by prior experience.

5.3.2 Subjects and Apparatus

28 people with different background experience and knowledge of smartphones or handheld AR participated in the user study. The participants age ranged from from 23 to 38 years. The distribution between the sexes was balanced with 12 female and 16 male participants. 21 users report to be adept in using smarthones and 16 report to play handheld 3D games. Furthermore, one person indicated in the pre-questionnaire to have occasionally severe pain issues in her/his primary hand's wrist. Nevertheless, all participants completed all 5 phases of both studies so that all data could be used for analysis.

The practical experiment is carried out using a Samsung Galaxy S II, as stated in Section 4.1.1. To minimise the risk of users pressing unintentionally the home- or power-button of the device, and therefore interrupting the experiment, the device was protected with a commercially available hard cover.

5.3.3 Statistical Foundation

The data collected in the user study is systematically analysed in a statistical evaluation to verify the hypothesis that were raised for the selection techniques in Section 5.4.1 and for the manipulation techniques in Section 5.5.1. The results of the evaluation are presented in Section 5.4.3 and 5.5.3. The evaluation was carried out using repeated measures one way ANOVA to detect significant differences in the results mean values.

ANOVA The analysis of variance (ANOVA) for repeated measures is a method to verify significant differences of two or more mean values of a given variable measured repeatedly. In this evaluation, the measured variables are the quantitative and qualitative data gathered for each investigated interaction technique, e.g. the completion time of a given task for each selection technique. To detect a significant difference, the F-test is used [21].

F-test The F-test is used to detect differences between the measurements of k related groups at a given significance level α and tests the following hypothesis:

$$H_0 =: \mu_1 = \mu_2 = \mu_3 = \dots = \mu_k$$

$$H_A =: \text{at least two means are significantly different}$$

and its results are expressed by the two values $(F_{m,n,p})$, with the degrees of freedom $m = k - 1$ and $n = \#samples - 1$. The F value, also called the F -ratio, expresses the ratio of systematic variation to unsystematic variation of the compared groups. The p value

is a probability, calculated with the F cumulative distribution function, describing how likely it is that a random sample, given the distribution of the observed variable, would have means farther or equally far apart as discovered for the tested sample. Generally, a high F value and a small p value indicate that H_0 can be rejected. However, to actually reject H_0 , the F -value has to be greater than the F -value of the inverse F cumulative distribution function for the given significance level α and the same degrees of freedom m, n .

As the F-test can only be used to find out that at least two means are significantly different in a group of k means, it is necessary (for $k > 2$) to perform pairwise comparisons of the different groups, to find out which groups differ significantly, using t-tests. Furthermore, the F-test and the t-test cannot be used for ordinal or ranked data, as e.g. from a Likert scale [26], at which Friedman's χ^2 test and the Wilcoxon signed-rank test are used as a replacement [12, 21].

Paired t-test The paired t-test is used to reject the null hypothesis H_0 that the difference of the mean values $d = \mu_1 - \mu_2$ is zero with:

$$\begin{aligned} H_0 &=: d = \mu_1 - \mu_2 = 0 & t_{n-1} &= \frac{\bar{x}}{s/\sqrt{n}} \\ H_1 &=: d \neq 0 & & \end{aligned} \quad (5.1)$$

its result is expressed by the tuple (t_{n-1}, p) , again indicating a rejection of H_0 if the value t_{n-1} is high and p is low. The value t_{n-1} is calculated with Equation 5.1 stated above as a fraction of the arithmetic mean \bar{x} and the standard deviation s for the combined variable $d_i = x_{1,i} - x_{2,i}$ for n paired samples. The null hypothesis can be rejected if the calculated t -value exceeds the t -value of the inverse cumulative function for the chosen significance level α and the same degrees of freedom $n - 1$ for the t -distribution [21].

Holm–Bonferroni Adjustment When employing paired tests to identify which two groups of k means differ significantly, the probability of type I errors (a mistakenly rejected null hypothesis) increases, as a family of hypothesis are verified in the same test. The Holm–Bonferroni adjustment [16] addresses this problem and can be used to control the family wise error rate. The algorithm for the Holm–Bonferroni adjustment works as follows. Instead of verifying each hypothesis H_1, \dots, H_m against a chosen significance level (e.g. $\alpha = 0.05$), the significance level is adjusted for each hypothesis. First, all p -values are calculated using an appropriate test method, in this work t -tests but Wilcoxon signed-rank tests for ordinal data, and order these ascendently. Now the hypothesis are tested, in order of significance, against their adjusted significance level α_k computed according to Equation 5.2.

$$\alpha_k = \frac{\alpha}{m + 1 - k}, \text{ with } k \in \{1, \dots, m\} \quad (5.2)$$

As such the significance level is most conservative at $\alpha_1 = \frac{\alpha}{m}$ for the hypothesis with the highest significance and is sequentially lowered till $\alpha_m = \alpha$ for the hypothesis with the least significant p -value. Once a hypothesis is rejected, all following hypothesis are rejected as well [16].

5.4 Selection

The *DrillSample* selection technique is evaluated in a user study according to Section 5.3 against the selection techniques *Ray-casting* and *Expand*, as outlined in Section 3.4.2. The three selection techniques are tested in three tasks that cover variations in object density and visibility.

5.4.1 Design & Objectives

The goal of the experiment is to evaluate the performance and ease-of-use of *DrillSample* compared to competing techniques. The focus of this study is on selection of objects in closer range in dense environments. A second objective is to examine the performance of the spatial context preservation of the proposed algorithm in environments with objects of high visual similarity. Table 5.5 holds the post-questionnaire to raise qualitative information, the hypotheses for the experiment are listed in Table 5.4.

- H_1 Ray-Casting will be best suited for non-occluded objects.
- H_2 Expand and DrillSample will perform considerably better than Ray-Casting in environments with overlapping, partly occluded or invisible objects, which differentiate significantly in appearance, in terms of speed and precision.
- H_3 Expand will suffer in environments with objects of high visual similarity. Likewise, DrillSample will perform considerably better than Expand in terms of speed and precision.

Table 5.4: Selection Hypothesis’.

- Q_1 How adequate do you feel the time allotted for practice was?
- Q_2 How comfortable were you with using a smartphone for task completion?
- Q_3 How would you rate the RAYCAST selection technique in terms of usability? Speed? Accuracy?
- Q_4 How would you rate the EXPAND selection technique in terms of usability? Speed? Accuracy?
- Q_5 How would you rate the DRILLSAMPLE selection technique in terms of usability? Speed? Accuracy?
- Q_6 Rank the three selection techniques in order of desired use (with 1 being the most desired).
- Q_7 When determining how much you like using a selection technique, how important in influence on your decision was usability? Speed? Accuracy?
- Q_8 Regarding the visualisation during the refinement process of the DRILLSAMPLE technique, how helpful and useful was the linear arrangement for spatial visualisation?

Table 5.5: Selection Technique Post-Questionnaire.

5.4.2 Test Scenarios

To cover different selection situations in dense 3D space, three different tasks are built. They range from unique and un-occluded to non-distinguishable and fully occluded object selection tasks. Thus, occlusion and visual similarity are used as variables for task design. As the underlying building block [32] for interaction design, the canonical task “selection” is applied, which refers to the task of acquiring a particular object from the entire set of objects available.

All tasks are based on the same virtual working ground (black & white textured plane) that was printed to paper at 56x40cm and acted as a marker for the Vuforia framework. The marker was placed on a table that was positioned at the center of a room so that users had around 150cm of obstacle free space to work within. All 28 users completed all three tasks in random order.

Task 1: Unique Object & No Occlusion The user was challenged to select a green cube in the middle of the working ground which was cluttered with around 80 other cubes of the same size but of different color (see Figure 5.13). The targeted object was easy to distinguish and not occluded by any of the objects in the scene. As soon the user selected or confirmed the selection of the green cube, the task finished automatically.

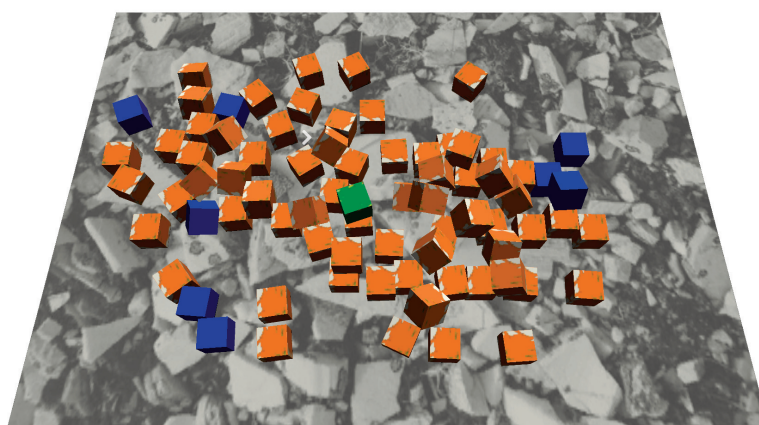


Figure 5.13: Task 1: “Select the green cube.”

Task 2 Unique Object & Strong Occlusion The user had to select a green brick in the lower right corner of a wooden textured box (see Figure 5.14). The box contained four stacks of different colored equally sized bricks. The targeted object was located on the very bottom of the last stack and it was the only brick that was colored in green. Although it was easy to distinguish, it was hardly visible due to the strong occlusion of the bricks stacked on top of it and the box’s walls. Again, on selection of the targeted object, the task finished automatically.

Task 3: Non-Distinguishable & Strong Occlusion In this task, the user had to select a brick from a wooden textured box again (see Figure 5.15). The box contained four stacks of equally sized bricks. All bricks were colored in light blue except for the bricks of the second stack

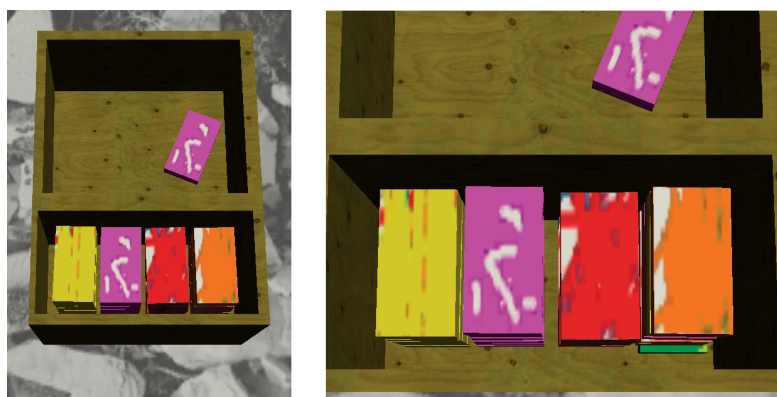


Figure 5.14: Task 2: “Select the green cube.”

which had a magenta colored texture. The targeted object was located on the very bottom of the magenta colored stack. It was only distinguishable by its position in the stack and was hardly visible due to strong occlusions of the bricks stacked on top of it and the box’s walls. The number of bricks on top of the targeted object varied randomly for each participant from four to seven pieces.

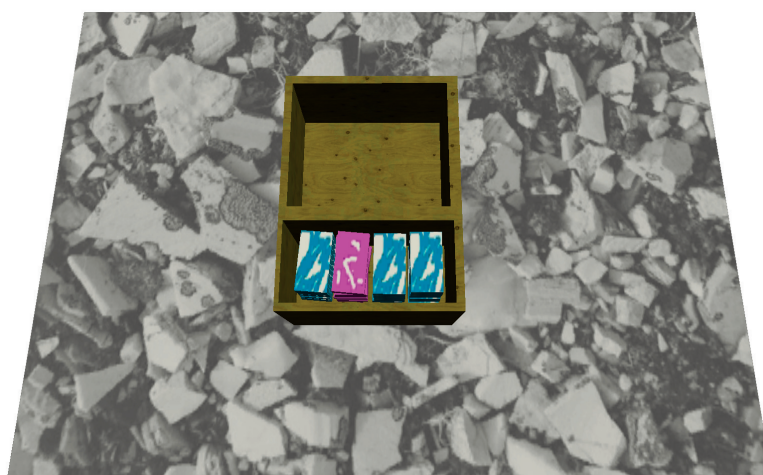


Figure 5.15: Task 3: “Select the lowermost of the pink bricks.”

5.4.3 Results

Based on the experiment’s results, an evaluation of the quantitative data to examine performance of the three techniques and a subjective evaluation regarding user’s preferences and feedback is conducted.

The quantitative data gathered from the questionnaires and automatically collected data of the test application were analysed with Friedman’s χ^2 test and repeated measures single factor

ANOVA accordingly. When suitable, pairwise t-tests or Wilcoxon signed rank test with the Holm’s sequential Bonferroni correction are applied. The focus lies on two different aspects during data analysis. First, data of all participants regarding selection techniques is evaluated and second, the analysis of the techniques’ performance depending on the specific tasks.

Evaluating the quantitative data, *Task Completion Time* and *Number of Selection Steps* were applied as metrics. Task completion time represents the time it takes to successfully finish a specific task from the time, the user started it. Number of selection steps comprises the amount of necessary object selections to successfully finish a selection task. This measure indicates precision of the applied technique.

5.4.3.1 Quantitative Evaluation

The evaluation of the completion time shown in Figure 5.16 indicates significant differences for the three interaction techniques with ($F_{2,54} = 6.74, p < 0.00243$) for all tasks on average but also with ($F_{2,54} = 9.27, p < 0.00035$), ($F_{2,54} = 21.84, p < 1.1e^{-7}$) and ($F_{2,54} = 4.91, p < 0.011$) for the tasks one to three separately. The pairwise t-test shows that only DrillSample is significantly faster than Ray-Casting with ($t_{27} = 4.33, p < 0.00018$) in the overall mean completion time.

For task 1, the techniques Ray-Casting and DrillSample score significantly better than Expand with ($t_{27} = -3.82, p < 0.0007$) and ($t_{27} = 2.65, p < 0.0134$). Most likely because Expand uses a cone-cast to select objects, which results more often in a refinement-step compared to DrillSample that casts a ray. No significant difference was measured between Ray-Casting and Drill-Sample. For task 2, the techniques with an additional refinement step prove to be faster than Ray-Casting with Expand at ($t_{27} = 7.8545, p < 1.9e^{-8}$) and DrillSample at ($t_{27} = 3.73, p < 0.0009$), however, no significant difference between DrillSample and Expand could be found. Here, Ray-Casting forces the user to successively select and put objects away until the desired object is easily accessible, which results in a very time-consuming problem. In task 3 it took users significantly less time to complete the task when using DrillSample, compared to Ray-Casting ($t_{27} = 3.24, p < 0.0031$) or Expand ($t_{27} = 2.6, p < 0.0148$). Ray-Casting fails as it did in task 2 because both problems force the user to move objects out of view step by step. Expand scores much worse than in task 2 because the targeted object cannot be distinguished from its spatial context and because Expand is only aligning the objects on a two dimensional grid. Between Ray-Casting and Expand no significant difference could be found.

Significant differences can be seen in Figure 5.17 for the results of the number of selections for task 2, 3 and on average, each with ($F_{2,54} = 10.98, p < 0.0001$). Task 1 shows no significant differences at ($F_{2,54} = 0.491, p < 0.615$) and advises that all selection techniques perform well in the simplest case.

The pairwise comparison for selection steps on average found the techniques Expand ($t_{27} = 15.29, p < 8.04e^{-15}$) and DrillSample ($t_{27} = 18.83, p < 4.7e^{-17}$) to be significantly better than Ray-Casting, but no significance among another at ($t_{27} = 1.31, p < 0.2$).

Similar to the task completion time, the number of selection steps in task 2 were significantly smaller for Expand at ($t_{27} = 18.4512, p < 7.78e^{-17}$) as well as for DrillSample with ($t_{27} = 13.93, p < 7.55e^{-14}$) compared to Ray-Casting, but also Expand ($t_{27} = -2.2, p < 0.036$) appears to be slightly less error-prone than DrillSample. Expand benefits in this task

from the fact that the targeted object is easily distinguishable, but also from its coarse selection volume where techniques casting a ray may have a hard time to hit an object that is only slightly visible. In task 3, likewise for average completion time, DrillSample is found having less false selections than Ray-Casting ($t_{27} = 16.87, p < 7.29^{-16}$) and Expand ($t_{27} = 2.61, p < 0.0146$). Additionally, Expand is significantly better than Ray-Casting at ($t_{27} = 8.34, p < 6.01^{-9}$), too. A possible cause for Expand scoring worst in terms of completion time, but not on number of false selections could be that each refinement step costs extra time for the visualisation, but also allows users to accidentally choose the targeted object each time.

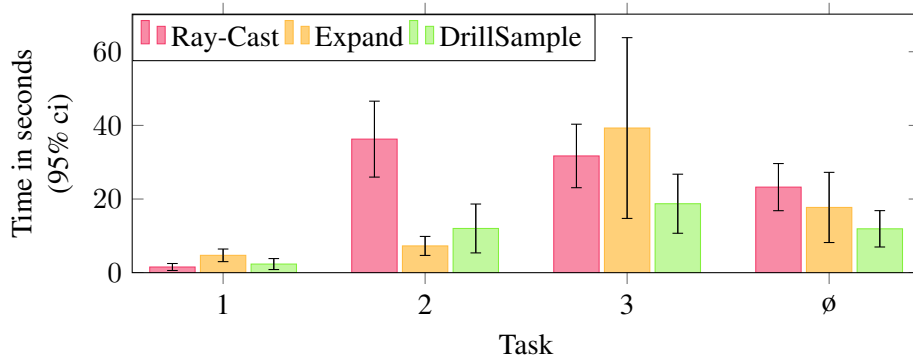


Figure 5.16: Mean completion time per task and on average.

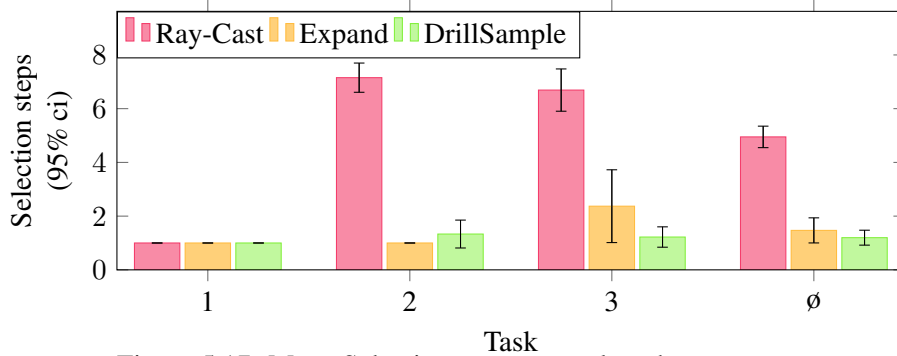


Figure 5.17: Mean Selection steps per task and on average.

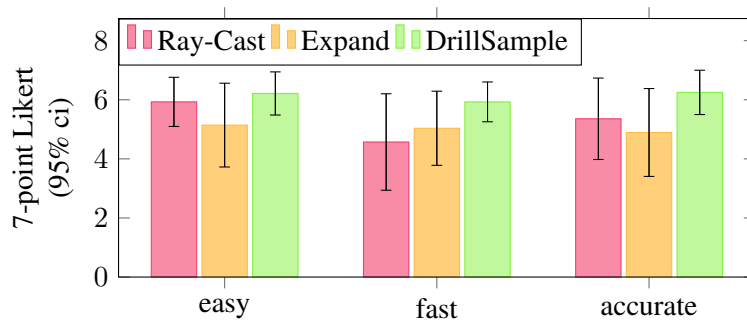
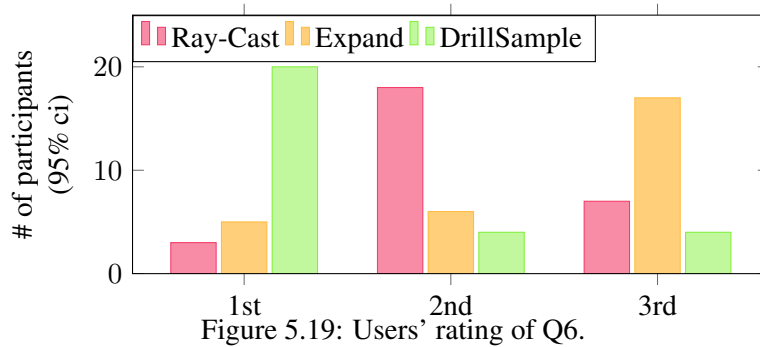


Figure 5.18: Users' average rating of Q3, Q4 and Q5.



5.4.3.2 Subjective Evaluation

Besides the performance measures based on quantitative data, the user's subjective evaluation on speed and accuracy of each technique is examined as well. Furthermore, the abstract performance value "ease-of-use" [6] to further evaluate the capabilities of the underlying technique is included. When answering the questions Q1-Q5, Q7 and Q8, users were able to choose from a 7-point Likert scale [26]. While all questions feature the highest rating at seven, and the lowest at one, Q1 states the best rating with four (appropriate).

The participants found the time allotted for practice appropriate with ($\mu = 3.93$ and $\sigma = 0.25$ at $\alpha = 0.05$). Using a smartphone to complete the different tasks was rated to be moderately comfortable with ($\mu = 5.72$ and $\sigma = 0.98$ at $\alpha = 0.05$). As depicted in Figure 5.18, all three techniques were rated at least above average but with significant differences regarding speed ($\chi_2^2 = 10.48, p < 0.0053$), ease-of-use ($\chi_2^2 = 9.53, p < 0.0085$) and accuracy ($\chi_2^2 = 15.27, p < 0.00048$).

Speed Only DrillSample was found to be significantly faster than Expand in the pairwise comparison ($W = -2.63, p = 0.0085$). Due to the Bonferroni adjustment, Ray-Casting failed to be significantly faster than Expand with ($W = -2.088, p = 0.0368$). Ray-Casting was not found to be significantly different from DrillSample ($W = -1.0558, p = 0.29108$). Expand was likely rated lower than the other techniques because it triggers refinement too often, while DrillSample only asks for refinement if objects overlap. Using Ray-Casting, users are not interrupted by a refinement step and might therefore consider it faster.

Ease of Use Users' ratings on ease-of-use found DrillSample significantly better than Ray-Casting and Expand at ($W = -2.84, p < 0.0045$) and ($W = 2.91, p < 0.0036$). Ray-Casting was insignificantly different to Expand with ($W = -0.89, p = 0.371$) even without the Bonferroni adjustment. Similarly, users found DrillSample significantly more accurate than Ray-Casting ($\chi_2^2 = 2.69, p < 0.007$) and Expand ($W = -3.17, p < 0.0015$). Likewise, Ray-Casting showed no significant difference to Expand at ($W = -1.23, p = 0.218$). Both Ray-Casting and Expand are not accurate if objects are occluded or look very similar. Hence, both factors result in a tedious, and when using Expand, even a confusing sequence of interactions to select the desired object.

Order of Desired Use For question Q6, asking the participant to rank the selection techniques in order of desired use, significant rankings for 1st ($\chi_2^2 = 18.5, p < 9.6^{-5}$) 2nd ($\chi_2^2 = 12.29, p < 0.0021$) and 3rd ($\chi_2^2 = 9.91, p < 0.007$) could be found, as shown in Figure 5.19. Rank one was clearly given to DrillSample with ($\chi_2^2 = -3.54, p < 0.00039$) and ($\chi_2^2 = -3, p < 0.0027$) significantly outranking Ray-Casting and Expand. Rank two was given to Ray-Casting with ($\chi_2^2 = -2.98, p < 0.0028$) and ($\chi_2^2 = -2.45, p < 0.014$) significantly outranking DrillSample and Expand. Rank three seems to be given to Expand, however, it only significantly outranks DrillSample with ($\chi_2^2 = -2.83, p < 0.0046$) but not Ray-Casting at ($\chi_2^2 = -2.04, p = 0.041$) due to the Bonferroni adjustment. All other pair-wise interaction technique tests show no significant difference.

Users stated all aspects of Q7 evenly important with 6 (important) or higher when answering Q6. Addressing in Q8, how helpful the spatial visualisation is, the participants found it useful to very useful with ($\chi_2^2 = 6.5$ and $\sigma = 0.1$ at $\alpha = 0.05$).

5.4.4 Qualitative Evaluation

Based on the 3D formalisation principles by Bowman and Hodges [6], a number of factors for the interaction task “3D selection” that influence performance in virtual environments were outlined. Since all three evaluated selection techniques are suited or explicitly designed for dense environments, “density” as a performance factor is not included. The specified factors are:

1. Object Size: This object property is related to the geometric area, a 3D object covers on the output device screen. A selection technique must be capable to select objects of varying size.
2. Occlusion: In any mixed reality environment, but especially in a dense environment, objects can partially or fully occlude each other which may result in invisible objects. In such environments, selection must be precise and provide some assisting visualisation to identify occluded objects.
3. Visual Appearance: The visual appearance of virtual objects can be of high similarity. Identifying the desired target object can result in problems in dense environments with occluded objects. In such environments, selection must provide an assisting visualisation to disambiguate the desired object.

Based on the results from quantitative as well as subjective evaluation, the findings with respect to object size, occlusion and visual appearance are summarised in Table 5.6.

Previous work [4, 10] report that Ray-Casting performs badly for objects covering only a small portion of the screen, while Expand performs well for the same case by casting a volume instead of a single ray. Beyond that, the findings indicate that Ray-Casting is well-suited for selecting non-occluded objects which can be also similar in appearance. However, if the desired object is small and is located amongst similar looking objects, imprecise touch input can evoke wrong selection. Compared to Ray-Casting, Expand is well suited to select visible or fully occluded objects of varying size. But the grid representation during the refinement step does

not provide full spatial correspondence to the original position of the selected objects; hence, precise selection of an object from a set of similar looking objects can be difficult and can result in wrong selections. DrillSample also lacks accuracy when selecting small objects due to the underlying use of Ray-Casting in combination with the imprecise single touch input. However, since DrillSample selects all objects which are cast by the ray, overlapping or occluded objects can be precisely selected due to DrillSample’s refinement step. Here, spatial context preservation provides a full overview that allows object disambiguation, which is especially of interest when selecting from a set of similar looking objects.

| | Object Size | Occlusion | Appearance |
|-------------|----------------------------------|-----------|------------|
| Ray-Casting | − _[10] _[4] | − | ○ |
| Expand | + _[10] | + | − |
| DrillSample | − | + | + |

Table 5.6: Summarisation of Results.

5.5 Manipulation

For a comprehensive evaluation of the proposed manipulation techniques, a summative evaluation across four different tasks with varying manipulation tasks is conducted.

5.5.1 Design & Objectives

The main goal of the experiment is to evaluate the performance and usability of 3DTouch and HOMER-S. Since 3DTouch matches the separated structure of the multi-touch input device and HOMER-S adapts real-world metaphors and incorporates integral 6DOF manipulation, both techniques apply for straightforward manipulation. Hence, a second objective is to compare both techniques and to examine intuitive handling. Table 5.7 lists the hypothesis’ that are formulated while designing the experiment and Table 5.8 holds the post-questionnaire (see Figures 7.9ff) to raise qualitative information for the manipulation techniques.

5.5.2 Test Scenarios

To cover different realistic manipulation situations in 3D space four different test tasks are built.

Bowman [4] considers the tasks *selection*, *position* and *rotation* as basic canonical tasks in spatial rigid transformations. However, selection has already been covered in a separate test and as *scaling* can be useful in different applications, the test will cover *position*, *rotation* and *scaling* instead. Using these basic tasks as building blocks [32], four tasks of varying complexity to allow for adequate simulation of manipulation in an AR application are designed. To manually identify the desired object for subsequent manipulation, another canonical task “selection” is used. Since the selection task is performed by all users in the same way and is equally designed

- H*₁ 3DTouch and HOMER-S are both designed to provide straightforward manipulation. Thus, both techniques will perform similar in terms of speed and ease-of-use for 3DOF manipulation tasks.
- H*₂ Since HOMER-S offers integral 6DOF transformation, it will perform considerably faster than 3DTouch in compound translation and rotation manipulation tasks.
- H*₃ Humans are able to control their single fingers more precisely than moving an object, which is attached to a device, freely in space. Thus, 3DTouch performs better for manipulation tasks that require precise input.
- H*₄ Regarding prior knowledge, users with experience using multi-touch devices will perform equally or better with 3DTouch than with HOMER-S. Likewise, the design of HOMER-S enables better performance for users with no prior multi-touch knowledge.

Table 5.7: Manipulation Hypothesis'.

- Q*₁ How adequate do you feel the time allotted for practice was?
- Q*₂ How comfortable were you with using a smartphone for task completion?
- Q*₃ How would you rate the 3DTouch manipulation technique in usability? Speed? Accuracy?
- Q*₄ How would you rate the HOMER-S manipulation technique in usability? Speed? Accuracy?
- Q*₅ How would you rate intuitiveness of 3DTouch for 2D-translate, 3D-translate, rotate, move & rotate, scale an object?
- Q*₆ How would you rate intuitiveness of HOMER-S for 2D-translate, 3D-translate, rotate, move & rotate, scale an object?
- Q*₇ Which manipulation technique do you prefer to 2D-translate, 3D-translate, rotate, move & rotate, scale an object?
- Q*₈ Rank the two manipulation techniques in order of desired use (with 1 being the most desired)?
- Q*₉ When determining how much you like using a manipulation technique, how important in influence on your decision was ease-of-use? Speed? Accuracy?

Table 5.8: Manipulaton technique post-questionnaire.

over all four tasks, the required time for selection does not influence the mean values of the performance metrics. The test was conducted under the same circumstances as the selection tests in Section 5.4.2.

Task 1: Positioning on a Plane The first task comprises the canonical task “positioning”. The user was challenged to translate a pink cube in the lower left corner to the center of a green area in the upper right corner (see Figure 5.20). The distance between the targeted object and its destination was 35cm on the horizontal plane. It was sufficient to complete the task with the cube partly overlapping the designated target.

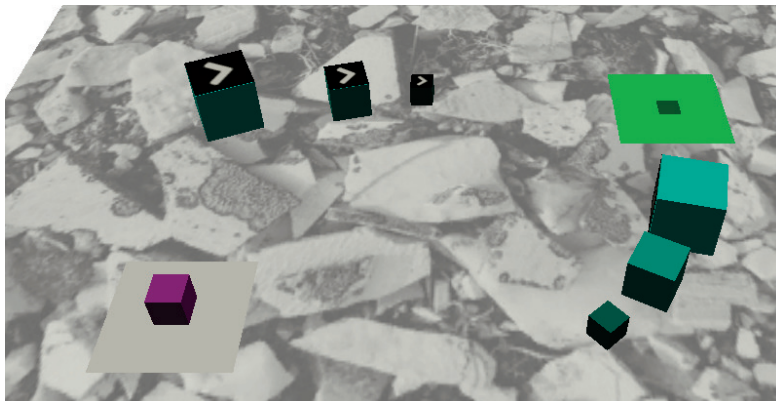


Figure 5.20: Task 1: “Move the *pink cube* to the center of the green & blinking platform”.

Task 2: Positioning in 3D Space The second task extends the first task by requiring positioning in all three dimensions. The user was challenged to translate a pink cube in the lower left corner on top of a small tower in the upper right corner (see Figure 5.21). The distance between the targeted object and its destination was 35cm on the horizontal plane and 20cm vertically. The destination area was again a square. If it was partly overlapped by the target object, the task was completed.

Task 3: Positioning & Rotation For better simulation of manipulation requirements in AR applications, an integral task design for the third task comprising a combination of “positioning” and “rotation” is applied. The user is challenged to rotate a red barrel in the lower left corner by 45° around its vertical axis and translate it on top of an inclined plane (see Figure 5.22). From there, the barrel was supposed to roll down the plane and over a square at its bottom. The test was successfully completed if the barrel was let loose on the top of the inclined plane rolling down its full length and at least partly hitting the center of the destination area.

Task 4: Scaling & Positioning A second integral task was designed for the fourth task. Here, the user was first challenged to scale a blue cube by a fifth in length and a third in width of its original size and then translate the cube into a glass positioned at the center of the scene (see

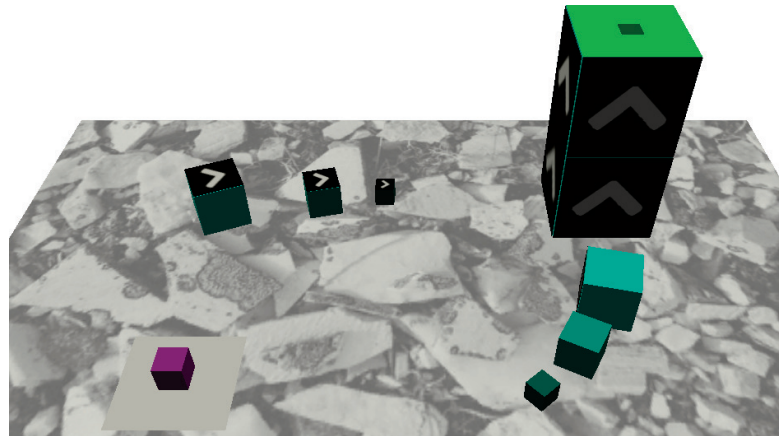


Figure 5.21: Task 2: “Move the *pink cube* on top of the green & blinking platform”.



Figure 5.22: Task 3: “Let the *barrel* roll down the inclined plane onto the green platform”.

Figure 5.23). The distance between the targeted object and its destination was 38 cm horizontally and 10 cm vertically. The destination was the circular shaped bottom of the glass. Users needed to let the cube fall into the glass from above and as soon as it hit the bottom, the task was completed.

5.5.3 Results

Based on the experiment’s results, an evaluation of the quantitative data to examine performance of the two techniques and a subjective evaluation regarding user’s preferences and feedback is conducted.

The quantitative data gathered from the questionnaires and automatically collected data of the test application were analysed with Friedman’s χ^2 test and repeated measures single factor ANOVA accordingly. The focus lies on three different aspects during data analysis. First, data of all participants regarding manipulation techniques is evaluated. Second, the techniques’ performance depending on tasks and third, the data of selected participants according to their

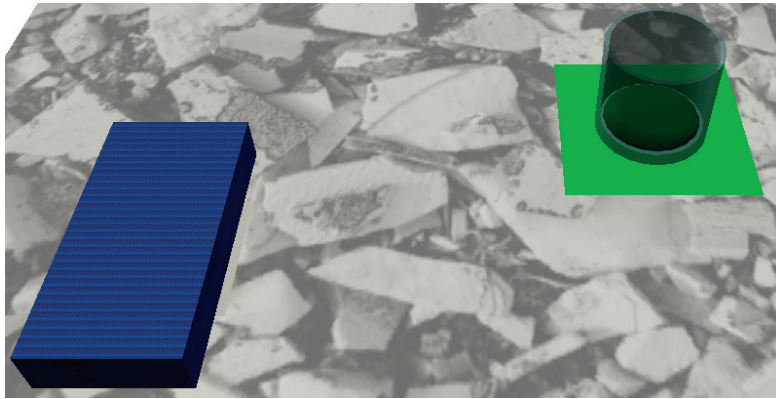


Figure 5.23: Task 4: “Resize and drop the *blue cube* into the glass”.

experience listed in Table 5.3 is analysed for each manipulation technique and task separately.

To evaluate the quantitative data, *Task Completion Time* and *Number of Interaction Steps* were the main evaluation criteria. Task completion time represents the time it takes to successfully finish a specific task. Number of interaction steps comprises the amount of necessary mode switches to successfully finish an (integral) manipulation task.

5.5.3.1 Performative Evaluation

Analysing the overall mean completion time, no significant difference was found between 3DTouch and HOMER-S ($F_{1,27} = 0.00299, p = 0.957$), as illustrated in Figure 5.24. When inspecting the mean completion time for each task separately, again no significant differences could be found for both positioning tasks of task 1 and 2 at ($F_{1,27} = 1.4, p = 0.2468$) and ($F_{1,27} = 0.814, p = 0.375$) respectively. However, the positioning & rotation task was performed significantly faster with HOMER-S ($F_{1,27} = 7.379, p < 0.0114$). In contrast to that, HOMER-S took significantly more time to complete for the scaling & positioning task ($F_{1,27} = 7.379, p < 0.0114$) according to Figure 5.25.

Analysing the task completion time, grouped by users’ knowledge according to Table 5.3, revealed no further significant differences other than the overall ones illustrated in Figure 5.25. No significant differences could be found for both positioning tasks when analysing the users’ experience. In the positioning & rotation task, the significantly better performance of HOMER-S was never independent of the users’ experience. The inexperienced users of the handheld gamer group (a) as well as of the smartphone group (b) performed significantly faster with HOMER-S than with 3DTouch. The experienced users of both groups performed faster with HOMER-S as well, but not significantly. Furthermore, only the experienced groups of a) and b) had significant results for task 4, since they were significantly faster using 3DTouch. No significant difference in performance between 3DTouch and HOMER-S could be found for the inexperienced users of both groups in task 4.

The results of the evaluation for the overall mean number of interaction steps exposed that 3DTouch enabled users to perform manipulations in significantly less steps than HOMER-S ($F_{1,27} = 4.552, p < 0.0421$), as illustrated in Figure 5.24. However, the evaluation per tasks

found only a significant difference in the Positioning in 3D Space task ($F_{1,27} = 4.374, p < 0.046$) and in the scaling & positioning task at ($F_{1,27} = 12.81, p < 0.0013$), both in favor of 3DTouch. Figure 5.26 indicates that there was no significant difference for the positioning on a plane or the positioning & rotation task with both ($F_{1,27} = 0.685, p < 0.415$).

The evaluation of mean number of interaction steps, grouped by users' experience revealed, with one exception for task 3, no deviant results than those illustrated in Figure 5.26. The significantly better performance of 3DTouch in the positioning in 3D space task could only be confirmed for the experienced users in a) and b). For the positioning & rotation task, the inexperienced group of a) achieved significantly better results with HOMER-S than with 3DTouch. For all other groups no significance could be found for that task. For task 4, only the experienced users of both groups had significantly better results with 3DTouch than with HOMER-S. No significant difference could be found for the inexperienced users of both groups.

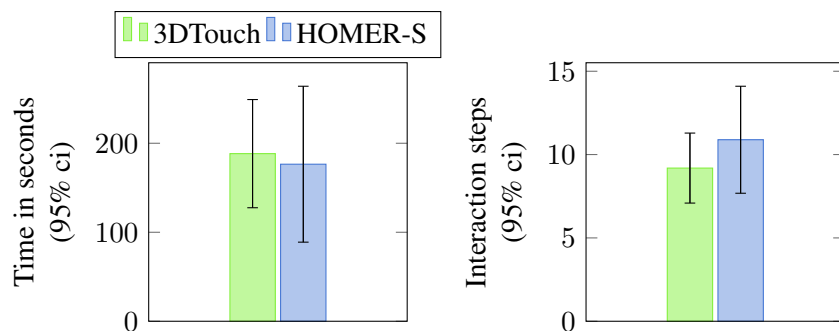


Figure 5.24: Mean completion time and number of interactions.

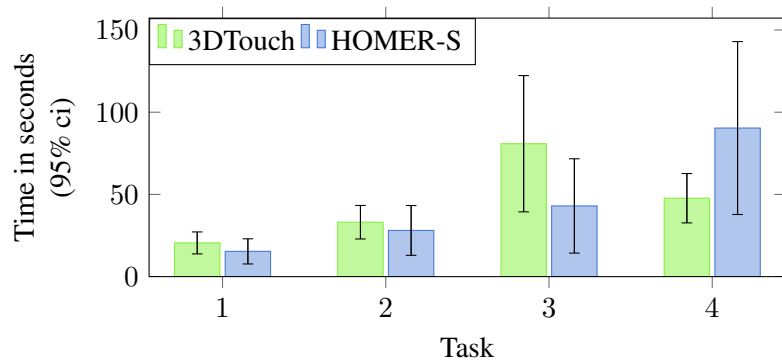


Figure 5.25: Mean completion time per task.

5.5.3.2 Subjective Evaluation

When answering the questions Q1-Q6 and Q9, users were able to choose from a 7-point Likert scale. While all questions feature the highest rating at seven, and the lowest at one, Q1 states the best rating with four (appropriate). The participants found the time allotted for practice appropriate ($\mu = 4, s = 0.46, a = 0.05$). Using a smartphone to complete the different tasks was rated to be moderately comfortable ($\mu = 5.9, s = 1.14, a = 0.05$).

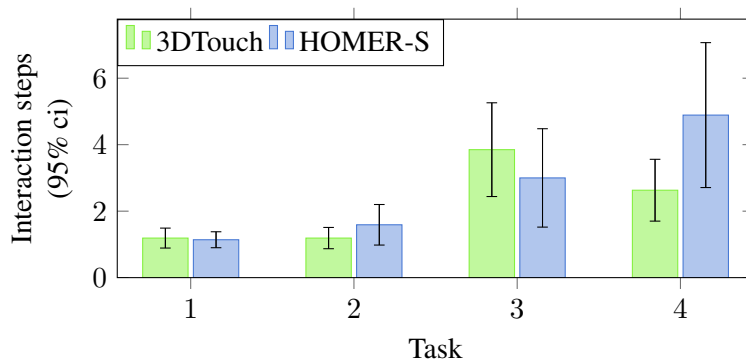


Figure 5.26: Mean number of interactions per task.

Usability, Speed & Accuracy As illustrated in Figure 5.27, the questions Q3 and Q4 revealed both to be average or good, but 3D Touch was rated significantly better for ease-of-use and accuracy with ($\chi^2[n : 28] = 6.55, p < 0.0105$) and ($\chi^2[n : 28] = 15.696, p < 0.0000744$) respectively. In terms of speed, no difference was confirmable. Analysing the subjective evaluation of ease-of-use, speed and accuracy, grouped by the user's experience, revealed significant better ratings of 3D Touch in ease-of-use only of the experienced users of a) and b). 3D Touch's better rating for accuracy was independent of the users experience except for inexperienced users in b) where no significant difference occurred.

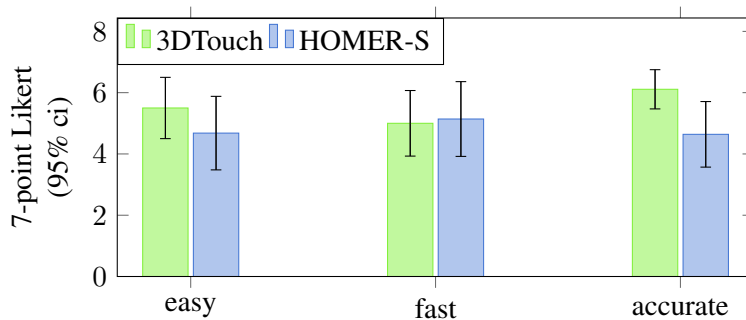
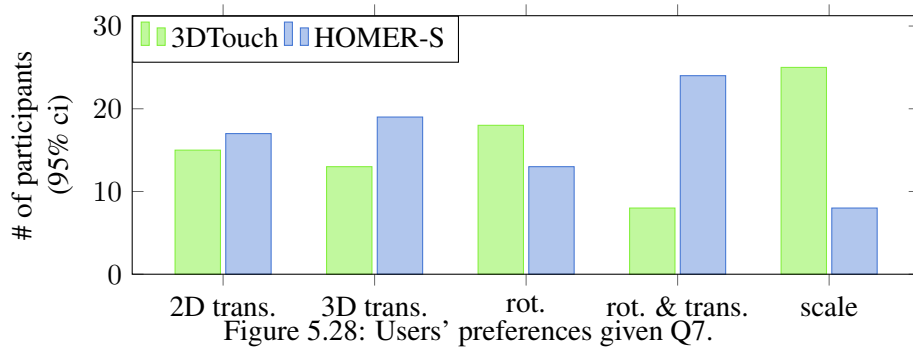


Figure 5.27: Users' average rating of Q3 & Q4.

Ranking & Preferences Users' ranking of the two interaction techniques indicated no significant preference (Q8) ($\chi^2[n : 28] = 0.57, p = 0.45$). Inspecting their preferences for specific transformations separately (Q7), a deeper insight could be gained. 2D- and 3D translation as well as rotation alone were not found to be significantly different, as shown in Figure 5.28. For the integral 6DOF manipulation of the positioning & rotation task, HOMER-S is significantly preferred with ($\chi^2[n : 28] = 10.67, p < 0.0011$). For scaling, 3D Touch is significantly preferred with ($\chi^2[n : 28] = 12.57, p < 0.00039$). This subjective evaluation reflects the results of the quantitative evaluation in terms on completion time. No deviant results for 2D- and 3D-translation as well as rotation alone were revealed, when analysing the ranking of each manipulation, grouped by the users' experience. Looking in detail at the users' preference of both groups for integral rotation and translation tasks, HOMER-S was significantly preferred by

the experienced users. Also the inexperienced users preferred HOMER-S, but not significantly. 3DTouch's preference for scaling remains independent of user's experience in both groups.



Question Q9 inquiring the users' influence on their decision for questions Q3 and Q4 yields with ($\chi^2[n : 28] = 3.89, p < 0.143$) no significant difference for the three options ease-of-use, speed and accuracy. Users stated all aspects of Q9 similarly important, ranging from $\mu = 5.5$ (slightly important) to $\mu = 6.18$ (important).

Conclusion & Outlook

In this chapter, the realised applications, interactions and findings of this thesis are summarised. Furthermore, it outlines open research topics that are worth to investigate in the future.

6.1 Modeling

As part of this thesis a modeling prototype for one-handed handheld augmented reality environments is developed. The prototype allows the effortless creation of geometric polygonal models using the *extrusion* and *lathe* modeling techniques on shapes that are drawn by hand with pen and paper. Models are instantly created and manipulable in real-time with simple one and two-finger gestures. By mapping a technique's parameters to the model's coordinate system's axes, the prototype abolishes iconic or textual menus for interaction. The parameters become accessible from different perspectives and the visible space of the virtual environment through the small screen of the handheld device is maximised.

The *ARTiFICe* framework is extended by a layer for the distribution of user generated content amongst participants in a collaborative environment. The distribution layer uses Unity's built-in asynchronous RPC functions to efficiently distribute full model meshes, model generation descriptions and model primitives.

As part of the modeling prototype, a simple, performant and robust application for the real-time detection of hand-drawn non-complex polygons without holes is implemented. It introduces the *push-to-find* interaction to simplify the capturing process and to motion blur, caused by tapping the screen that would affect the detection negatively. Important factors, like poor ambient illumination that influence the robustness and accuracy of the detection pipeline are identified and optimised by noise reduction and adaptive thresholding.

To enable communication between the modeling application and the shape detection application, a framework for *Remote Method Invocation* is developed that focuses on ease-of-use and cross-platform interoperability. It enables effortless local and remote inter-process communication using TCP/IP for any application written in either JAVA or C#.

The *perspective driven modeling* approach has three drawbacks that should be addressed in the future that would improve its performance and usability.

1. Currently, the polygons that are provided by the shape detection application are triangulated using *ear trimming* [30]. This approach is simple but is likely to produce very unbalanced triangulations with narrow angled triangles. A future implementation should therefore use an algorithm for triangulation that produces optimal triangulations to improve the quality of the generated meshes.
2. The meshes that are generated by the modeling application are fully re-created if a parameter of the used modeling technique is changed. As a result, currently the manipulation of large models becomes infeasible in real-time. Thus, the routines that generate a model's mesh should reuse the result of previous generations.
3. The *perspective driven modeling* approach did not undergo an iterative design cycle, so that matters regarding its intuitivity relies mostly on the author's personal experience. Therefore, the modeling application should be subject to an evaluation with users to identify usability problems.

The user interface and the image processing pipeline of the shape detection application suffer of two known usability problems and a limitation in its application.

1. The application for shape detection is configured to detect the shapes as precise and fast as possible. However, choosing the right image resolution that should be processed currently relies on empirical data. To get a good user experience it is important to choose the right image resolution for different reasons. It should be as high as possible to be able to capture shapes with the highest possible accuracy. Furthermore, it must be small enough, so that the pipeline is able to process it in real-time. And third, it should not be much lower than the device's screen resolution to provide an accurate and sharp AR view. A future implementation should therefore be able to choose optimal settings given the handheld's hardware specifications.
2. The polygon approximation in step 9 of the processing pipeline approximates a shape's contour to retrieve a coarser outline. Depending on the shape's characteristic (angled, round), a different level of approximation may be desired. Currently it is tiresome to change the approximation level as it has to be changed in the application's settings dialog. It would be much more convenient to interactively adjust the level once the shapes have been detected.
3. So far only non-complex polygons without holes are accepted for further modeling. A future implementation should allow the use of complex polygons.

6.2 Selection

A user study is designed to compare three different techniques in terms of speed, precision and ease-of-use for performing 3D selection tasks with a multi-touch handheld device in a dense AR

scene. Many of the outcomes of the performance study are statistically significant and allowed to draw multiple meaningful conclusions. In H1, Ray-Casting is proposed to be best suited for selection of non-occluded objects. Results of completion time for task 1 support H1, since Ray-Casting significantly outperforms Expand. H1 can further be strengthened by taking the subjective evaluation into account where users considered Ray-Casting to be fast. DrillSample also performed significantly better than Expand for task 1. This indicates the strength of techniques casting a ray instead of casting a cone for visible object selection in close range, since a ray selects fewer objects. Thereby, just a few objects need to be presented at DrillSample's refinement step, while Cone-Casting is always coarser. There, more objects are presented during a refinement step, which takes more time for a user to get an overview before indicating the desired object. Therefore H1 can be supported to be true in terms of speed. Regarding precision, neither performance, nor subjective evaluation revealed statistical significance to back up H1. Therefore, H1 does not hold in terms of precision.

Results for evaluating speed and precision when selecting almost fully occluded objects clearly reveal Expand's and DrillSample's strengths. Both perform significantly faster and need less selection steps than Ray-Casting, which supports H2. Since no significant difference in completion time and interaction steps between Expand and DrillSample could be found, H2 can be backed up further. These results indicate that Expand and DrillSample are both equally suited for selecting an occluded object, which highly differs in appearance from the surrounding ones. Regarding precise selection of occluded objects with high visual similarity, DrillSample significantly outperforms both baseline techniques in terms of completion time and number of interaction steps. Based on these results, H3 can clearly be supported. It proves the advantage of the proposed spatial context preservation compared to the grid representation that Expand provides. The disadvantage of Expand's detailed visualisation becomes even more apparent, since no significant difference in completion time could be found between Expand and Ray-Casting.

Regarding users' preference, the subjective evaluation clearly reveals users' being in favor of DrillSample. It significantly outranked both baseline techniques when users are asked for an overall ranking. This first rank can further be confirmed when looking at the details. Users ranked DrillSample highest in terms of speed, precision and ease-of-use. It significantly outperformed Expand in terms of speed, but not Ray-Casting. Since Ray-Casting does not provide a refinement step, it tends to be considered fast and "direct". The DrillSample's capability to precisely select the desired object over all three test scenario is ranked significantly best in terms of precision. Finally, the users ranked DrillSample significantly best in ease-of-use.

Based on these results and findings, a set of preliminary guidelines regarding object selection in closer range could be derived:

1. Ray-Casting remains a good alternative selection technique, as long as objects are fully visible.
2. Expand remains a good alternative for visible or occluded objects of varying object size, as long as they differ in visual appearance.
3. For visible or occluded objects, independent of their visual appearance, DrillSample is the best general purpose method.

6.2.1 Outlook

As part of this work 3D selection techniques in handheld AR environments are explored by evaluating three techniques. The main motivation is precise selection of objects in dense one-handed handheld AR. Therefore, it is intended to reduce multi-touch input due to implicit restrictions having only one hand available for selection. DrillSample requires only single touches as input and splits up the procedure into two steps. The spatial context is preserved if multiple objects have been indicated as targets, to allow for disambiguation and precise selection of occluded objects or objects with high similarity in visual appearance.

The performance study clearly revealed the strengths of the DrillSample technique compared to related work in precise selection of objects in dense environments within close range. Although DrillSample was tested in handheld AR, the technique applies for dense VEs as well. As future work various object density and distance combinations will be explored. It is planned to investigate using DrillSample with Cone-Casting to provide accurate selection of smaller objects at a larger distance. Furthermore, performance and usability of DrillSample is to be examined when selecting objects of varying size. Therefore, a closer look is taken on competing techniques like SQUAD [23], Expand [10] and Dual-Finger Selection Techniques [51] for latter evaluation.

6.3 Manipulation

A user study is designed to compare two different techniques for performing 3D manipulation tasks with a multi-touch handheld device. While 3DTouch separates the DOFs of the task to improve performance as shown in previous work [29], HOMER-S controls 6DOF in an integral way and takes advantage of simulating real-world metaphors.

Results show that for both techniques, no significant difference is found for overall mean task completion time, completion time for the positioning tasks, overall user preference or user preferences regarding the positioning tasks. These results support hypothesis H1 and confirm the straightforward, intuitive design of both interaction techniques when performing canonical tasks. Inspecting performance and user's preference for compound canonical tasks, two findings can be stated. First, for 6DOF manipulation tasks, as simulated in the positioning & rotation scenario, HOMER-S performed significantly faster than 3DTouch. This quantitative evaluation is supported by the user's subjective feedback. HOMER-S is significantly preferred for translation and rotation tasks by users as expressed in Q7. These findings support H2 and indicate the strength of the integral design of HOMER-S for compound canonical 6DOF tasks. This is also reflected by users' comments who described HOMER-S to be natural, of "more direct contact" and fun. Thus, these real world metaphors tend to be very intuitive and straightforward. The second finding when inspecting performance and user's preference for composite manipulation tasks reveals the strength of 3DTouch for scaling tasks. It took considerably less time to complete the scaling & positioning task using 3DTouch than with HOMER-S. Furthermore, users significantly preferred 3DTouch for scaling. Since no significant difference is found regarding the positioning tasks in completion time or user preferences, positioning can be neglected when evaluating scenario 4. This finding supports H3, since the scaling tasks required very fine ma-

nipulation in all three dimensions. H3 can further be backed up by the significant fewer number of interaction steps 3DTouch needed in tasks 2 and 4 and the users' rating in Q3 and Q4 attested it a better accuracy.

Besides the assumption that humans are able to control their fingers more precisely, the underlying metaphor can be another conceivable reason to further explain the underperformance of HOMER-S in scaling tasks. In the real world, usually two hands are involved to expand or shrink an object. Since HOMER-S only provides one virtual hand to simulate one real hand, this metaphor could not be adapted in a direct way. Thereby, a direct mapping could not be provided that limits HOMER-S straightforward usage for scaling. However, the pinch-like gesture to scale an object using 3DTouch is also not completely intuitive and straightforward. Since, more than half of the test group classified themselves as experienced handheld 3D gamers, they are familiar with using multi-touch for interaction; standard touch gestures such as the pinch-out and -in are known and well trained. This is also backed up by the results including user experience. There, 3DTouch results for scaling are only significantly better for users who are experienced with smartphones or handheld 3D gaming. Studying further details regarding user experience leads to H4. It is proposed that prior touch knowledge would result in equal or better performance of 3DTouch compared to HOMER-S, while inexperienced users would perform better with HOMER-S due to its integral 6DOF design and adaption of real-world metaphors. For many results of the study, this is true. Regarding completion time, no significant differences between 3DTouch and HOMER-S could be found for positioning when analysing experienced users. For 3D positioning, experienced users needed significantly less interaction steps when using 3DTouch. For integral positioning and rotation, experienced users of both groups performed faster with HOMER-S, but not significantly. Experienced users performed significantly faster for scaling in terms of completion time and number of interaction steps when using 3DTouch. They rated 3DTouch significantly better in terms of ease-of-use, but significantly preferred HOMER-S for 6DOF transformation. Regarding inexperienced users, H4 can be further backed up by the significant better performance in terms of completion time and number of interaction steps for the positioning & rotation task using HOMER-S. Users' comments reflect the quantitative results. Most users, especially the inexperienced, reported to have quickly familiarised with HOMER-S for any translations and rotations. However, exceptions when evaluating H4 could be found, too. The quantitative results do not indicate a better performance of inexperienced users using HOMER-S for positioning tasks. For scaling, HOMER-S did not result in better performance of the inexperienced users. However, despite of the good results of 3DTouch for scaling, inexperienced users did not significantly perform better using 3DTouch for scaling. The underlying two-fingers pinch gesture requires prior knowledge and thus, is not as straightforward and direct than the one-finger inputs for translate and rotate. But users' preference of 3DTouch's for scaling is independent of the users' experience. This is also reflected by users' comments. Some users experienced HOMER-S as being "too direct", since even small body movements result in a manipulation. Most users complained about HOMER-S being unintuitive to use for scaling. Based on these observations, a clear conclusion to support H4 cannot be drawn. Further research needs to be performed for a detailed evaluation of this hypothesis.

Taking the results and findings into account, a set of preliminary guidelines is developed:

1. Both methods provide intuitive manipulation with similar performance when the canonical

manipulation tasks “positioning” and “rotations” are required.

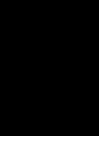
2. *HOMER-S* outpaces *3DTouch* in performance and ease-of-use to perform compound positioning and rotation tasks.
3. *3DTouch* is the better choice, if scaling is involved in the manipulation task.

6.3.1 Outlook

As part of this work, a foundation for exploring 3D manipulation techniques in one-handed handheld AR environments is provided by evaluating two manipulation techniques. The main motivation is the development of intuitive manipulation techniques for 3D content in handheld AR. Therefore, the design aimed at reducing or even eliminating (multi-) touch input to avoid abstraction levels, which require prior knowledge. A novel multi-touch manipulation technique *3DTouch* is introduced, which provides 6DOF transformations by separating degree-of-freedom and combining one- and two-finger touch input with the device pose. Thereby, necessary touch input could be reduced to a minimum. This improves usability for a one-handed handheld AR environment by utilizing only simple finger gestures; single touches for translation and rotation and two-finger input for scaling. Additionally, the manipulation technique *HOMER-S* is presented that completely decouples object transformation from limited handheld screen space. It does not require any touch input, but uses only the device pose to provide full 6DOF transformations.

The evaluation study clearly reveals the strengths of both techniques depending on given manipulation tasks. Based on the promising results of both techniques, the performance and usability of a combination of *3DTouch* and *HOMER-S* will be examined. The findings on fine manipulation tasks motivate to further evaluate the capabilities of both techniques. Additionally, it is planned to optimise the overall usability of the *HOMER-S* manipulation technique to further exploit its potential. The focus will be to improve pose stability during manipulation and to improve the gesture for scaling. To overcome stated rotation limitations using the *HOMER-S* approach, a non-direct mapping between device pose and object’s rotation will be evaluated.

CHAPTER 7



Appendix

Selection in Handheld Mixed Reality

Welcome

Thank you very much in participating into this user study.

The study comprises 3 parts and will take approx. 20 minutes for completion:

1. Pre questionnaire
2. Practical exercise: Selection in Handheld MR
3. Post questionnaire

We will request only your gender and age as personal information. Your answers to the questions of the pre- and post questionnaire will be stored and analyzed anonymously.

1. What is your gender?

- Female
- Male

2. How old are you?

years

Please proceed to the pre-questionnaire by hitting the "Next" button

Selection in Handheld Mixed Reality

Pre-Questionnaire

Please answer all of the following questions, then proceed to the practical exercise.

3. About how often do you play video games?

- Never Once a Year A few time a Year Once a Month Every Week Every Day
-

Figure 7.1: Selection technique questionnaire part 1/6.

4. What % of your gaming is playing mobile 3D games?

Please specify: 0 10 20 30 40 50 60 70 80 90 100

5. Do you have a multi-touch Smartphone?

- Yes
- No
- No, but I regularly use one.

6. Do you have any experience with mobile Augmented Reality?

None I used once a mobile AR app I regularly use mobile AR apps I develop mobile AR apps I have no or little mobile AR experience, but I have experience with Virtual Reality applications

7. Where do you rank your general 3D spatial skill?

Poor Below Average Average Above Average Excellent

8. Do you have any flexibility or pain issues with your primary hand, fingers or arm?

- No
- Yes. Please specify the pain issue.

Figure 7.2: Selection technique questionnaire part 2/6

Selection in Handheld Mixed Reality

Practical Exercise

During the practical part, you will be asked to count virtual objects. Please write down the numbers using paper and pen.

Upon completion of the practical test, please proceed to the post questionnaire.

Selection in Handheld Mixed Reality

Information of the Practical Part

9. Please enter the your User ID:

User ID is provided at the end of the practical exercise on the mobile screen.

UserID

10. Please enter, for each selection technique separately, the number of bricks that had covered the lowermost pink brick:

| | | |
|--------------|----------------------|--------|
| RAYCAST | <input type="text"/> | Bricks |
| EXPAND | <input type="text"/> | Bricks |
| DRILL SAMPLE | <input type="text"/> | Bricks |

Figure 7.3: Selection technique questionnaire part 3/6.

Post-Questionnaire: Selection

Please answer all of the following questions.

11. How appropriate do you feel the time assigned for practice was?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Much too short | Moderately too short | Slightly too short | Appropriate | Slightly too long | Moderately too long | Far too long |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

12. How comfortable were you with using a mobile phone for task completion?

| | | | | | | |
|-----------------------|--------------------------|------------------------|-----------------------|-----------------------|------------------------|-----------------------|
| Very uncomfortable | Moderately uncomfortable | Slightly uncomfortable | Neither Nor | Slightly comfortable | Moderately comfortable | Very comfortable |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

13. How would you rate the RAYCAST selection technique in terms on how ...

| | | | | | | | |
|---|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| | Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| ... Easy it was to select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Fast you could select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Accurate the selection of a desired object was? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

14. How would you rate the EXPAND selection technique in terms on how ...

| | | | | | | | |
|---|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| | Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| ... Easy it was to select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Fast you could select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Accurate the selection of a desired object was? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 7.4: Selection technique questionnaire part 4/6.

15. How would you rate the DRILLSAMPLE selection technique in terms on how ...

| | Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
|---|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| ... Easy it was to select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Fast you could select a desired object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Accurate the selection of a desired object was? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

16. Rank the three selection techniques in order of desired use (with 1 being the most desired).

Please drag and drop the selection techniques to the ranks.

| | |
|--------------|--------------------------------|
| RAYCAST | <input type="text" value="1"/> |
| EXPAND | <input type="text" value="2"/> |
| DRILL SAMPLE | <input type="text" value="3"/> |

17. When determining how much you like using a selection technique, how important in influence on your decision was ...

... How easy it was to select a desired object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... How fast you could select a desired object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... How accurate the selection of a desired object was?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 7.5: Selection technique questionnaire part 5/6.

... The possibility to select hidden objects?

| | | | | | | |
|-------------------------|-----------------------|-----------------------------|-----------------------|---------------------------|-----------------------|-----------------------|
| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

18. Regarding the visualization during the refinement process of the DrillSample technique, how helpful and useful was ...

... linear arrangement for spatial visualization of selected objects?

| | | | | | | |
|-------------------------|---------------------------|-------------------------|-----------------------|------------------------|--------------------------|------------------------|
| Strongly Useless | Moderately Useless | Slightly Useless | Neither Nor | Slightly Useful | Moderately Useful | Strongly Useful |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... possibility to inspect selected objects (zoom, rotate)?

| | | | | | | |
|-------------------------|---------------------------|-------------------------|-----------------------|------------------------|--------------------------|------------------------|
| Strongly Useless | Moderately Useless | Slightly Useless | Neither Nor | Slightly Useful | Moderately Useful | Strongly Useful |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

19. Please state advantages and disadvantages you experienced when using DRILLSAMPLE technique for object selection.

Page 05
G02

20. Please feel free to add any comment about the study and the techniques.

Last page

Selection in Handheld Mixed Reality

Thank You!!!

We would like to thank you very much for helping us.

Annette Mossel, Benjamin Venditti, www.ims.tuwien.ac.at, Interactive Media Systems Group, Vienna University of Technology - 2012

Figure 7.6: Selection technique questionnaire part 6/6.

Transformation in Handheld Mixed Reality

Welcome

Thank you very much in participating into this user study.

The study comprises 3 parts and will take approx. 20 minutes for completion:

1. Pre questionnaire
2. Practical exercise: Transformation in Handheld MR
3. Post questionnaire

We will request only your gender and age as personal information. Your answers to the questions of the pre- and post questionnaire will be stored and analyzed anonymously.

1. What is your gender?

- Female
- Male

2. How old are you?

years

Please proceed to the pre-questionnaire by hitting the "Next" button

Transformation in Handheld Mixed Reality

Pre-Questionnaire

Please answer all of the following questions, then proceed to the practical exercise.

3. About how often do you play video games?

- Never Once a Year A few time a Year Once a Month Every Week Every Day
- -
 -
 -
 -
 -

Figure 7.7: Manipulation technique questionnaire part 1/6.

4. What % of your gaming is playing mobile 3D games?

0 10 20 30 40 50 60 70 80 90 100

Please specify:

5. Do you have a multi-touch Smartphone?

- Yes
- No
- No, but I regularly use one.

6. Do you have any experience with mobile Augmented Reality?

None

I used once a mobile AR app

I regularly use mobile AR apps

I develop mobile AR apps

I have no or little mobile AR experience, but I have experience with Virtual Reality applications

7. Where do you rank your general 3D spatial skill?

Poor

Below Average

Average

Above Average

Excellent

8. Do you have any flexibility or pain issues with your primary hand, fingers or arm?

- No
- Yes. Please specify the pain issue.

Transformation in Handheld Mixed Reality

Practical Exercise

Upon completion of the practical test, please proceed to the post questionnaire.

Figure 7.8: Manipulation technique questionnaire part 2/6.

Transformation in Handheld Mixed Reality

Information of the Practical Part

9. Please enter the your User ID:

User ID is provided at the end of the practical exercise on the mobile screen.

UserID

Post-Questionnaire

Please answer all of the following questions.

10. How appropriate do you feel the time assigned for practice was?

Much too short
 Moderately too short
 Slightly too short
 Appropriate
 Slightly too long
 Moderately too long
 Far too long

11. How comfortable were you with using a mobile phone for task completion?

Very uncomfortable
 Moderately uncomfortable
 Slightly uncomfortable
 Neither nor
 Slightly comfortable
 Moderately comfortable
 Very comfortable

12. How would you rate the MULTI TOUCH manipulation technique in terms on how ...

| | Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
|--|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| ... Easy it was to use? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Fast you could manipulate an object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Accurate the manipulation was? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 7.9: Manipulation technique questionnaire part 3/6.

13. How would you rate the DEVICE ORIENTATION manipulation technique in terms on how ...

| | Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
|--|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| ... Easy it was to use? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Fast you could manipulation an object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Accurate the manipulation was? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

14. Which manipulation technique do you prefer to ...

| | MULTI TOUCH | DEVICE ORIENTATION | I liked both the same |
|---|-----------------------|-----------------------|-----------------------|
| ... Move an object in two dimensions (i.e. on a plane)? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Move an object in three dimensions? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Rotate an object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Move and rotate an object at the same time? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| ... Scale an object? | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

15. Rank the two manipulation techniques in order of overall desired use (with 1 being the most desired).

Please drag and drop the manipulation techniques to the ranks.

MULTI TOUCH



DEVICE ORIENTATION



16. When determining how much you like using a manipulation technique, how important in influence on your decision was how ...

... Easy it was to use?

| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Fast you could manipulate an object?

| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Accurate the manipulation was?

| Very Unimportant | Unimportant | Slightly Unimportant | Neither Nor | Slightly Important | Important | Very Important |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 7.10: Manipulation technique questionnaire part 4/6.

17. How would you rate intuitiveness of the MULTI TOUCH technique for ...

... Moving an object in two dimensions?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Moving an object in three dimensions?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Rotating an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Moving and rotating an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Scaling an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

18. How would you rate intuitiveness of the DEVICE ORIENTATION technique for ...

... Moving an object in two dimensions?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Moving an object in three dimensions?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Rotating an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Moving and rotating an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

... Scaling an object?

| | | | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Extremely Poor | Poor | Below Average | Average | Above Average | Good | Excellent |
| <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 7.11: Manipulation technique questionnaire part 5/6.

19. When determining how intuitive a manipulation technique was for you, how important in influence on your decision was ...

... Prior knowledge of using multi touch interfaces?

Very Unimportant Unimportant Slightly Unimportant Neither Nor Slightly Important Important Very Important

... Possibility to manipulate an object (move, rotate, scale) by moving/rotating the mobile device?

Very Unimportant Unimportant Slightly Unimportant Neither Nor Slightly Important Important Very Important

20. Please state advantages and disadvantages you experienced when using DEVICE ORIENTATION technique for object manipulation.

Page 05
G02

21. Please feel free to add any comment about the study and the techniques.

Last page

Transformation in Handheld Mixed Reality

Thank You!!!

We would like to thank you very much for helping us.

Close window

Annette Mossel, Benjamin Venditti, www.ims.tuwien.ac.at, Interactive Media Systems Group, Vienna University of Technology - 2012

Figure 7.12: Manipulation technique questionnaire part 6/6.

Bibliography

- [1] Ronald T Azuma et al. A survey of augmented reality. *Presence*, 6(4):355–385, 1997.
- [2] Richard A. Bolt. “put-that-there”: Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '80, pages 262–270, New York, NY, USA, 1980. ACM.
- [3] Leonardo Boshell. LitJSON v0.9.0, JSON library for the .Net framework. <http://lbv.github.io/litjson>. [accessed 15-March-2014].
- [4] D. Bowman, E. Kruijff, J. LaViola Jr., and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison-Wesley, 2005.
- [5] Doug A Bowman and Larry F Hodges. An Evaluation of Techniques for Grabbing and Manipulating Objects in Immersive Virtual Environments Arm-Extension Ray-Casting. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 35–38. ACM, 1997.
- [6] Doug a. Bowman and Larry F. Hodges. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. *Journal of Visual Languages & Computing*, 10(1):37–53, February 1999.
- [7] Fabrice Boyer. TrueSculpt Virtual Sculpture. <https://play.google.com/store/apps/details?id=truesculpt.main>. Google Play, [accessed 15-March-2014].
- [8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.
- [9] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [10] Jeffrey Cashion, Chadwick Wingrave, and Joseph J LaViola. Dense and dynamic 3D selection for game-based virtual environments. In *IEEE Virtual Reality*, volume 18, pages 634–42, April 2012.
- [11] FasterXML and Codehaus. Jackson JSON Processor v2.1.5. <http://wiki.fasterxml.com/JacksonHome>. [accessed 15-March-2014].

- [12] Andy Field, Jeremy Miles, and Zoë Field. *Discovering Statistics Using R*. SAGE Publications, 2012.
- [13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
- [14] Sinem Guven, Steven Feiner, and Ohan Oda. Mobile augmented reality interaction techniques for authoring situated media on-site. In *IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 235–236. IEEE, October 2006.
- [15] Anders Henrysson, Mark Billinghurst, and Mark Ollila. Virtual object manipulation using a mobile phone. In *Proceedings of the 2005 international conference on Augmented tele-existence (ICAT '05)*, pages 164–171. ACM, 2005.
- [16] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.
- [17] Google Inc. Android Software Development Kit v4.4. <http://developer.android.com/sdk>. [accessed 15-March-2014].
- [18] Google Inc. Gestures, Android Developer Guide. <http://developer.android.com/design/patterns/gestures.html>. [accessed 19-October-2014].
- [19] Google Inc. Intents and Intent Filters, Android Developer Guide. <http://developer.android.com/guide/components/intents-filters.html>. [accessed 15-March-2014].
- [20] Google Inc. Services, Android Developer Guide. <http://developer.android.com/guide/components/services.html>. [accessed 15-March-2014].
- [21] Jürgen Janssen and Wilfried Laatz. Statistische datenanalyse mit spss. *Eine anwendungsorientierte Einführung in das Basissystem und das Modul exakte Tests*, 7, 2010.
- [22] Hannes Kaufmann. *Geometry Education with Augmented Reality*. PhD thesis, Institut für Softwartechnik und Interaktive Systeme, 2004.
- [23] Regis Kopper, Felipe Bacim, and Doug a. Bowman. Rapid and accurate 3D selection by progressive refinement. In *2011 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 67–74. IEEE, March 2011.
- [24] GA Lee, Ungyeon Yang, Y Kim, D Jo, and KH Kim. Freeze-Set-Go interaction method for handheld mobile augmented reality environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 143–146. ACM, 2009.
- [25] Jiandong Liang and Mark Green. Jdcad: A highly interactive 3d modeling system. *Computers & Graphics*, 18(4):499 – 506, 1994.

- [26] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [27] David Makofske, Michael J. Donahoo, and Kenneth L. Calvert. *TCP/IP Sockets in C#: Practical Guide for Programmers (The Practical Guides)*. Morgan Kaufmann, 2004.
- [28] Anthony Martinet, Gery Casiez, and Laurent Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115–118. IEEE, 2010.
- [29] Anthony Martinet, Géry Casiez, and Laurent Grisoni. Integrality and separability of multi-touch interaction techniques in 3D manipulation tasks. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):369–80, March 2012.
- [30] G. H. Meisters. Polygons Have Ears. *The American Mathematical Monthly*, 82(6):648–651, 1975.
- [31] Annette Mossel, Christian Schönauer, Georg Gerstweiler, and Hannes Kaufmann. ARTiFICe-Augmented Reality Framework for Distributed Collaboration. *International Journal of Virtual Reality*, 11(3):1–7, 2012.
- [32] M.E. Mündel. *Motion and Time Study: Improving Productivity*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc, 1978.
- [33] Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, and Morgan Skinner. *Professional C# 2012 and .Net 4.5*. John Wiley & Sons, 2012.
- [34] OpenCV. Open Source Computer Vision Library v2.4.9. <http://opencv.org>. [accessed 15-March-2014].
- [35] I. Poupyrev, T. Ichikawa, S. Weghorst, and M. Billinghurst. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. *Computer Graphics Forum*, 17(3):41–52, 1998.
- [36] Ivan Poupyrev and Mark Billinghurst. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80. ACM, 1996.
- [37] Qualcomm. Augmented Reality (Vuforia) v3.0. <https://developer.qualcomm.com/mobile-development/add-advanced-features/augmented-reality-vuforia>. [accessed 15-March-2014].
- [38] Qualcomm. Extending Unity Android Activity. <https://developer.vuforia.com/resources/dev-guide/knowledge-base-articles>. [accessed 15-March-2014].
- [39] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

- [40] Jason L. Reisman, Philip L. Davidson, and Jefferson Y. Han. A screen-space formulation for 2D and 3D direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST '09)*, pages 69–78, New York, New York, USA, 2009. ACM.
- [41] David F Rogers. *An introduction to NURBS: with historical perspective*. Morgan Kaufmann, 2001.
- [42] Mario Russo. *Polygonal Modeling: Basic and Advanced Techniques*. Jones & Bartlett Learning, 2006.
- [43] und Keiichi Abe. Satoshi Suzuki. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.
- [44] Scalisoft. Spacedraw. <https://play.google.com/store/apps/details?id=com.scalisoft.spacedraw>. Google Play, [accessed 15-March-2014].
- [45] Carsten Seifert. *Spiele entwickeln mit Unity: 3D-Games mit Unity und C# für Desktop, Web & Mobile*. Carl Hanser Verlag GmbH Co KG, 2014.
- [46] Sai Srinivas Sriparasa. *JavaScript and JSON Essentials*. Packt Publishing, 2013.
- [47] William Steptoe. Sketcher 3D. <https://play.google.com/store/apps/details?id=com.Doktor3D.Sketcher3D>. Google Play, [accessed 15-March-2014].
- [48] Richard Stoakley, Matthew J Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM Press/Addison-Wesley Publishing Co., 1995.
- [49] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.
- [50] Unity Technologies. Unity Game Engine v4.5. <http://unity3d.com>. [accessed 15-March-2014].
- [51] Can Telkenaroglu and Tolga Capin. Dual-finger 3d interaction techniques for mobile devices. *Personal and ubiquitous computing*, 17(7):1551–1572, 2013.
- [52] Manuel Veit, Antonio Capobianco, and Dominique Bechmann. Influence of degrees of freedom’s manipulation on performances during orientation tasks in virtual reality environments. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pages 51–58. ACM, 2009.
- [53] Daniel Wagner. *Handheld augmented reality*. PhD thesis, Citeseer, 2007.

- [54] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134. IEEE Computer Society, 2008.