Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Making Sense of Lines: Interaction Sequences for 3D Modeling with Mid-Air Sketches

Master's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Jan Benscheid

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Torsten Kuhlen

Registration date: 15.04.2019
Submission date: 15.10.2019

Eidesstattliche Versicherung

Benscheid, Jan	331427
Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Making Sense of Lines: Interaction Sequences for 3D Modeling with Mid-Air Sketches

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum	Unterschrift

Contents

List of Figures

List of Tables

Abstract

A distinctive property of 3D modelling in the field of Personal Fabrication is, that the models are created with the goal of materializing them eventually, for example using a 3D printer. Thus, the models often have a strong relation to their designated setting, e.g. by being supposed to be mounted on an object. The designing of such models is therefore often accompanied by frequent measurements.
In this context, it seems desirable to create the designs "in-situ", directly in their supposed location, with the aid of Augmented Reality (AR). Current smartphones are already capable of creating AR views that stay aligned with the physical world.

The ARPen is the implementation of a concept, with which in-situ modeling can be realized with comparatively accessible hardware. Next to the smartphone, a 3D-printed pen is used, which can be tracked by the app thanks to visual markers, and thereby act as a spatial pointing device.
The goal of this work was to investigate on the creation of 3D models under such conditions. Therefore, the ARPen system was enhanced by respective capabilities, in order to eventually use it in a user study to test various modeling techniques. On the basis of our literature research, we decided to choose our techniques from the field of solid modeling.

Our results yield first insights about the user's behaviour and preferences. Our participants highly valued techniques, which were most efficient. In particular, our findings once again demonstrate the importance of including physical surfaces into the design process. Precise unconstrained spatial input is impeded by numerous factors, like insufficient motor abilities and human's improper spatial perception in smartphone AR. The most appropriate input technique therefore heavily depends on the physical environment.

Überblick

Eine Besonderheit des 3D-Modellierens im Bereich *Personal Fabrication* ist, dass die Modelle mit dem Ziel gestaltet werden, sie schlussendlich — beispielsweise durch einen 3D Drucker — zu erstellen. Daher weisen sie häufig einen hohen Bezug zu ihrer vorgesehen Umgebung auf, beispielsweise indem sie an existierenden Objekten befestigt werden sollen. Das entwerfen solcher Modelle ist somit oftmals mit häufigen Messungen verbunden.
In diesem Kontext erscheint es naheliegend, unter Zuhilfenahme von *Augmented Reality* (AR), die Objekte unmittelbar an ihrem Bestimmungsort "in situ" zu modellieren. Aktuelle Smartphones sind in bereits der Lage, AR Ansichten zu erzeugen, welche an der realen Umgebung ausgerichtet sind.

Der ARPen ist die Umsetzung eines Konzepts, mit dem in situ Modellierung mit vergleichsweise leicht verfügbaren Mitteln umgesetzt werden könnte. Neben dem Smartphone kommt ein 3D-gedruckter Stift zum Einsatz, der sich dank visueller Marker innerhalb der App als räumliches Zeigegerät verwenden lässt.
Ziel dieser Arbeit war es, das Erstellen von 3D Modellen unter eben diesen Bedingungen zu erforschen. Dafür wurde das ARPen System um entsprechende Funktionalitäten erweitert, um schlussendlich mit dessen Hilfe in einer Nutzerstudie verschiedene Techniken zur Erstellung von Modellen zu testen. Auf Basis der Erkenntnisse unserer Literaturrecherche, haben wir uns in dieser Studie für Techniken aus dem Bereich des *Solid Modeling* mit Linienskizzen entschieden.

Unsere Ergebnisse geben erste Rückschlüsse auf das Nutzerverhalten und Präferenzen. Unsere Teilnehmer legten hohen Wert auf die Effizienz ihrer Eingaben. Insbesondere stellte sich erneut die Wichtigkeit heraus, vorhandene physikalische Oberflächen in den Designprozess zu integrieren. Präzise räumliche Eingaben werden von zahlreichen Faktoren wie unzureichenden motorischen Fähigkeiten und der eingeschränkten räumlichen Wahrnehmung des Menschen in aktuellen Smartphone AR systemen erschwert. Die geeignetste Eingabetechnik kann daher abhängig von der aktuellen physikalischen Umgebung stark variieren.

Acknowledgements

I'd like to thank Prof. Dr. Jan Borchers and Prof. Dr. Torsten Kuhlen for examining my thesis.

I thank my supervisor, Philipp Wacker, for his kind and helpful support throughout the entire time, and for his patience when our meetings got a little lengthy ever now an then.

Of course I also thank all people who participated in my user study.

Lastly, I want to thank the FabLab HiWis for their support in all FabLab related manners. I really had a great time there!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition: *Excursus*

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in American English and the generic feminine is used.

Download links are set off in coloured boxes.

File: myFile[a]

[a]http://hci.rwth-aachen.de/public/ARPen/file_number.file

Chapter 1

Introduction

With 3D printing becoming ever more accessible to the public due to decreasing prices, improved usability and the emergence of Fab Labs[1], people are able to manufacture many custom and downloaded designs. The creation of such designs however remains a complex task. Especially the modeling of objects with relation to the real world require measuring and spatial ability, in order to accurately picture the model in its desired surrounding.

In-situ modeling may ease the creation objects with relation to the real world.

Augmented reality (AR) enables the creation of modeling environments in which objects can be designed *in-situ*. One such implementation is the ARPen ([Wacker et al., 2019]). The system consists of a smartphone app which provides an augmented reality (AR) view, and a 3D-printed pen which is tracked by the app in space via visual markers. In this thesis, we investigate how 3D modeling can be done in this setting, both from a technical, as well as an interaction viewpoint.

Previous findings suggest that pen-like input devices—even in a 3D scenario—afford sketching lines ([Hoffmann, 2017]). However, the desired outcome of a 3D modeling process is usually an object with volume. From literature, we identified three common categories of approaches on how to handle the creation of volumetric geometry based on spatial line

[1]http://fab.cba.mit.edu/

Sketch input has to be converted into volumetric geometry.

input. Those are painting with brushes that produce strokes with surface or volume ([Schkolne et al., 2001], [Ponto et al., 2013]), creating curve networks and spanning faces across the gaps ([Wesche and Seidel, 2001], [Milosevic et al., 2015], [Jackson and Keefe, 2016]) and sweeping two-dimensional profiles into the third dimension ([Ara, 2012], [Weichel et al., 2014]) as it is common in solid modeling software.
Aside from the initial creation of objects, combining them can be useful as well. [Weichel et al., 2014] integrated Boolean operations into their mixed reality modeling environment and especially demonstrated their utility for integrating real-world objects into the design process.

Depth perception is a problem especially in smartphone AR.

The ARPen case offers additional benefits, but also challenges. The previous work mostly did not consider in-situ modeling, where using physical surfaces as guidance can highly improve the sketching performance ([Arora et al., 2017], [Wacker et al., 2018]). Also, contrary to systems using a head-mounted or stationary display, looking from different angles can be done with less body movement in the ARPen's bi-manual interaction style. Lastly, the ARPen concept could be available to all owners of a modern smartphone. On the other hand, the ARPen does not offer stereoscopic vision—-a potential source of precision errors ([Kruijff et al., 2010]). Previous experience with the ARPen has shown that users have trouble to create consistent mid-air sketches with an increase in complexity, often manifested as poor spatial alignment of the strokes. Also, the visual tracking of the pen through the phone's camera requires hand coordination.

In this work, we focus on solid modeling techniques.

Given the unique properties of the ARPen system, we approached the question of how users can create complex and custom 3D models in such a scenario. We chose to focus our investigations on modeling techniques from the solid modeling domain, as we considered the efficiency of the operations (in terms of line input required compared to the complexity of the resulting shapes) to be beneficial in the context of the ARPen.
For evaluating the modeling techniques with users, a working implementation would enable the participants to see and reflect upon the results of their inputs. One part of

this work was therefore dedicated to the implementation of a framework that enables the generation of non-trivial 3D geometry within the ARPen app. The second part was the exploration of interaction techniques. For this, we conceptualized and implemented a set of techniques which we used in the user study to gain insights on user's behaviour and preferences.

We implemented 3D modeling with the ARPen and tested it in a user study.

This thesis starts with an overview over the domain, accompanied by the definitions of important re-occurring terms (Chapter 2 "Background"). After that, there is a more in-depth review of related work (Chapter 3 "Related work"), where we cover immersive modeling, mid-air sketching and techniques in CAD modeling. Chapter 4 describes the structure of the library implementation for enabling the geometry creation. Based on our findings from Chapter 3, Chapter 5 "Modeling Techniques" is about the interaction techniques which we implemented for the user study, covering both conceptual as well as realization aspects. In the following chapter (Chapter 6 "User Study"), we explain the user study in detail, including the discussion of the results. We finally conclude the thesis with Chapter 7, in which we summarize our results and discuss potential areas for future work.

Chapter 2

Background

2.1 Mixed Reality

[Milgram and Kishino, 1994] famously depicted mixed reality as a section of the *Reality-Virtuality Continuum* (Figure 2.1). At the left end, there are fully real environments, independent of whether they are observed directly or through any kind of monitor. At the very end, there are solely virtual environments, which we would commonly refer to as virtual reality (VR). By the definition from [Milgram and Kishino, 1994], mixed reality therefore covers the part of the spectrum in which real-world and virtual objects are visible within the same display. In augmented reality (AR), virtual objects are overlaid with the real world. There are some common ways in which this

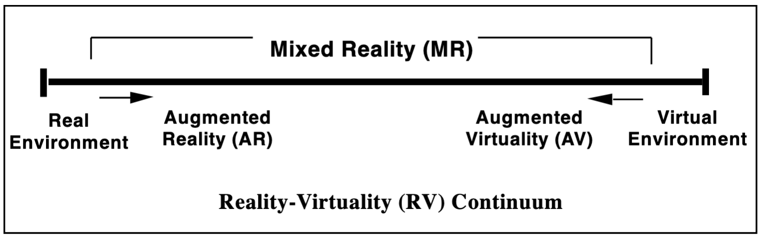


Figure 2.1: The simplified representation of the RV continuum according to [Milgram and Kishino, 1994]

can be realized:

There are various types of AR displays.

With *head mounted displays*, the display is kept constantly in the wearer's view. The display can be optical see-trough, meaning that the virtual content is displayed on a transparent pane (e.g. Microsoft HoloLens[1]) or video see-through, meaning that the wearer sees the real world on the opaque display through a camera (e.g. as possible with the Samsung Gear VR[2]).

Handheld displays can be carried by the user in their hands. Usually these devices are video see-through. This may also include applications in which a front-facing camera is used in an AR mirror fashion. Due to its convenience (it can be realized on smartphones, which are widely available) it is a very commonly used type of AR display.

Additionally, there are *stationary displays* and *projector* setups, where the virtual content is projected onto the real world at the appropriate place. *Cave setups* and the like, where the content is displayed on fixed monitors but perspectively adjusted to the viewer, represent an edge case between AR and VR. Here, real content does overlay virtual content. While this is an undersired artifact in some situations, it can also be used deliberately.

2.2 Digital 3D Modeling Landscape

In the following, we will give a brief overview over existing techniques for the creation of 3D models, that are commonly used in practice. For this work, we will categorize them by the type of primitives that the designer interacts with on a high level. Our categories are *polygonal modeling* (Section 2.2), *digital sculpting* (Section 2.2), *solid modeling* (Section. 2.2) and *other* (ref. 2.2 "Other").

For each category, we will give a short description including what metaphors it is based on and how manipulations are done, name its main areas of application and give examples for popular tools which are based on the respective techniques. Terminology that will reoccur through-

[1]https://www.microsoft.com/en-us/hololens
[2]https://www.samsung.com/global/galaxy/gear-vr/

out this work will be explained in detail at appropriate places. The information in this section is predominantly based on the book "Digital Modeling" by William Vaughan ([Vaughan, 2012]).

Polygonal Modeling

Polygonal is a popular technique in the gaming and animation industry. An example of a polygonal model is shown in Figure 2.3 left. 3D models are represented by a mesh data structure which the user manipulates directly by creating, removing and shifting primitives. Those primitives are:

Polygonal modeling is especially popular in the entertainment industry.

Vertices Points in 3D space.

Edges Lines connecting two vertices.

Faces Sets of faces forming closed polygons. From the designer's perspective, faces are usually planar and most often triangular or quadrangular.

When modeling in this style, a popular approach is to use a generic shape as the starting point and refine it until the desired mesh is acquired.
As the designer has direct control over the mesh topology—particularly the degree of detail—, this approach is well suited for areas in which rendering performance plays a crucial role. It also provides high flexibility regarding the nature of the shapes to be modeled, which can have both sharp and organic features. However, its focus on working with discrete geometry makes it less suited for precise engineering tasks. Also, mesh manipulation can be time consuming compared to more domain-specific modeling techniques. Modern polygonal modeling tools such as Blender[3] therefore tend to incorporate other techniques as well.

Polygonal modeling is flexible, but produces discrete geometry.

[3]https://www.blender.org/

Digital Sculpting

Digital sculpting is well-suited for modeling organic shapes.

Digital sculpting uses modeling with deformable materials such as clay as a metaphor ([De la Flor and Mongeon, 2010]). In contrast to polygonal modeling, the geometry's underlying data structure (i.e. polygonal meshes, voxels) is more or less transparent to the designer, depending on the software used. It is the software's responsibility to render the model into a mesh with the desired resolution at export time.
The user modifies objects with brushes that e.g. push, pull or carve the material. This makes digital sculpting especially well-suited for creating organic shapes such as characters (Figure 2.3 center). A practical benefit of this technique for rendering applications is, that fine details that would get lost during discretization can be preserved by rendering them into a texture.
However, this interaction style makes digital sculpting less applicable for objects with sharp corners and exact measures.

Solid Modeling

Solid modeling tools are currently highly popular in personal fabrication.

Solid modeling is the common approach in application areas like engineering, product design and architecture (Figure 2.3 center). As in these fields there are often strict specifications to be met, there is a high emphasis on precise measurements and fidelity. Regarding the field of personal fabrication, we found that among the 20 most popular items from the categories *Household* and *Gadgets* on Thingiverse[4] for which the used software was explicitly stated, 17 were created using solid modeling tools, the most prominent ones being Autodesk®Fusion 360[5] (8) and Tinkercad[6] (3).

[4]https://www.thingiverse.com/
[5]https://www.autodesk.de/products/fusion-360/overview
[6]https://www.tinkercad.com/

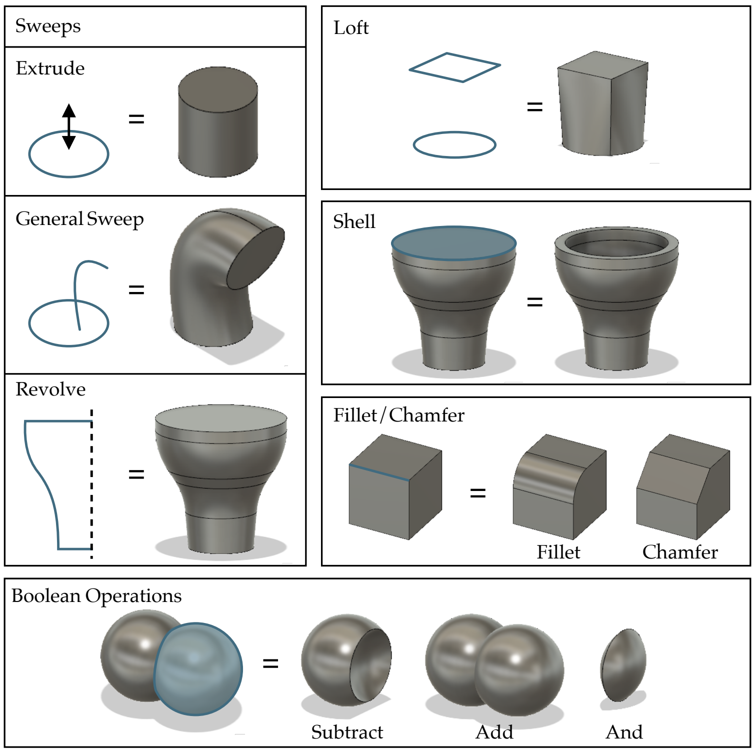


Figure 2.2: Overview of CAD operations.

Common modeling techniques in CAD software include (Figure 2.2) ([Li et al., 2010], [Culpepper]):

Sweeping Creating a volume by moving a profile along a trajectory. If the trajectory can be arbitrary, this is often referred to as a *general sweep*. For convenience, there exist common additional functions: A sweep, restricted to the profile's normal direction is often referred to as an *extrusion*. A circular sweep—analogous to a rotation around an axis—is a *revolution*.

Lofting Creating a shape that interpolates a set of profiles.

Fillet/Chamfer Refining edges by rounding or applying chamfers.

Shelling Removing faces from a solid object, while converting all remaining ones into a wall with thickness.

CSG Constructive solid geometry (CSG) allows combining objects using common Boolean operations such as *union*, *difference* and *intersection*.

While traditionally, interaction with CAD software used to be highly command-based, there is a movement towards *direct modeling*, where it is possible to shape and deform objects using drag and drop gestures ([Alba, 2018]).

Parametric approaches are useful for customizable and generative designs.

Besides many WYSIWYG modeling tools, there are also script-based editors like OpenSCAD[7], which follow a *parametric* approach. In the maker community, this is often used for customizable designs. Grasshopper[8] uses a visual flow-based programming environment in which—in combination with Rhino[9]—complex parametric designs can be realized. Antimony[10] is another CAD tool which relies solely on visual flow-based programming[11]. As it uses Functional Representation (F-Rep) for its geometry (contrary to Boundary Representation (B-Rep)), it enables certain operations which are hard to achieve in other CAD tools, such as blending between shapes.

Definition: *B-Rep, F-Rep*

B-Rep, F-Rep:
In solid modeling, Boundary Representation (B-Rep) means that the software internally represents volumetric geometry by its boundary surface(s). In Functional Representation (F-Rep), volumes are defined by equations that classify each point in space as inside- or outside the shape.

[7]https://www.openscad.org/
[8]https://www.rhino3d.com/6/new/grasshopper
[9]https://www.rhino3d.com/
[10]https://www.mattkeeter.com/projects/antimony
[11]Users can add custom script nodes which are coded in Python.

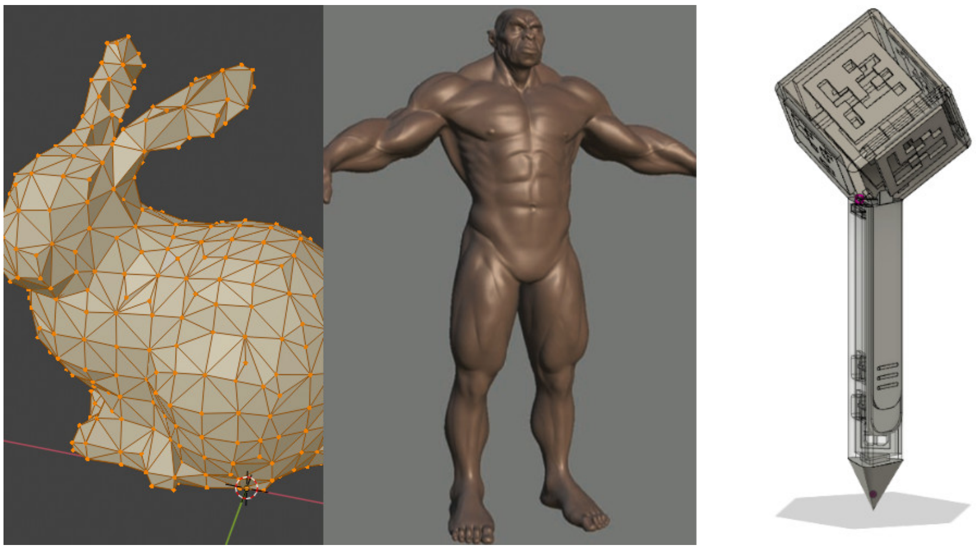


Figure 2.3: Screenshots from polygonal modeling (left), digital sculpting (center) [De la Flor and Mongeon, 2010] and solid modeling (right).

Other

With *3D scanning*, instead of building a model digitally from ground up, a real-world object is scanned and digitized. Aside from simply scanning existing objects, there have also been approaches to use the technology for creative purposes. [Anderson et al., 2000] used 3D scanning to digitize clay models for a tangible modeling experience. [Weichel et al., 2015] extended this concept by enabling a back and forth between the physical and the digital model. Using an additive and a constructive tool head, the designer's changes to the model can be applied interactively. [Weichel et al., 2014] combined 3D scanning with an AR modeling environment, making it possible to easily incorporate real-world objects into the design process.

3D scanning can be used to incorporate real-world geometry into the design.

Another more recent approach is the use of machine learning in modeling tasks. A typical use case is the retrieval of models from a database, that match a user's sketch ([Wang et al., 2015], [Eitz et al., 2012], [Su et al., 2015]).

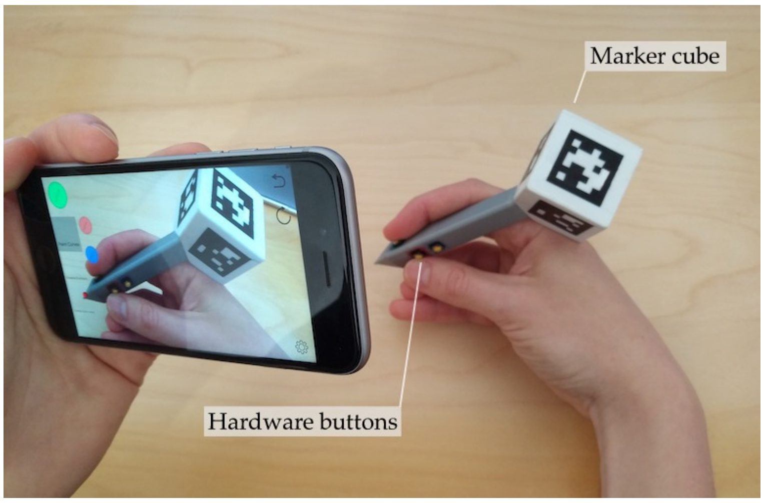


Figure 2.4: The ARPen model used in our study.

2.3 ARPen

The ARPen is an implementation of a bi-manual smartphone AR concept.

The ARPen system ([Wacker et al., 2019]) consists of a 3D-printed pen and a smartphone app (Figure 2.4). The app uses Apple's ARKit to create an augmented reality view with a coordinate system aligned with the real world. The pen has a cube with visual markers at its end, which enables the app to track it with six degrees of freedom (6DoF) if at least one of the markers is visible for the device camera. Among other applications, this can be used to infer the position of the pen tip in space and thereby use the pen as a 3D pointing device. Furthermore, the state of the pen's hardware buttons is transferred to the app via a Bluetooth connection. The system is designed to be operated holding the phone in one hand and the pen in the other. Depending on the grasp, this enables operating parts of the phone's touch screen with the phone hand.

Chapter 3

Related work

Evaluating a very diverse set of modeling techniques—possibly even non-sketch-based ones—would have certainly been interesting. However, we considered the implementation effort to be disproportionate for the scope of this work. We therefore decided to focus our attention on sketch-based modeling techniques, as previous findings suggest the high affordance of sketching for pen-shaped input devices, even in a mid-air modeling scenario ([Hoffmann, 2017]). For this work, we focused on techniques for modeling from scratch. Note, that there is also a large body of research regarding retrieval-based approaches (ref. Section 2.2 "Other").

We focus our research on sketching-based approaches.

This chapter will start with a review of past projects in the field of immersive modeling, focusing on sketching-based methods, followed by a section about the caveats of mid-air sketching. Afterwards, we will go more into detail regarding the operations in CAD modeling, and present different input gestures from related work.

3.1 Immersive 3D Modeling

The origins of immersive 3D modeling date back to at least 1992, where [Butterworth et al., 1992] presented a modeling system consisting of a 6DoF input device and VR

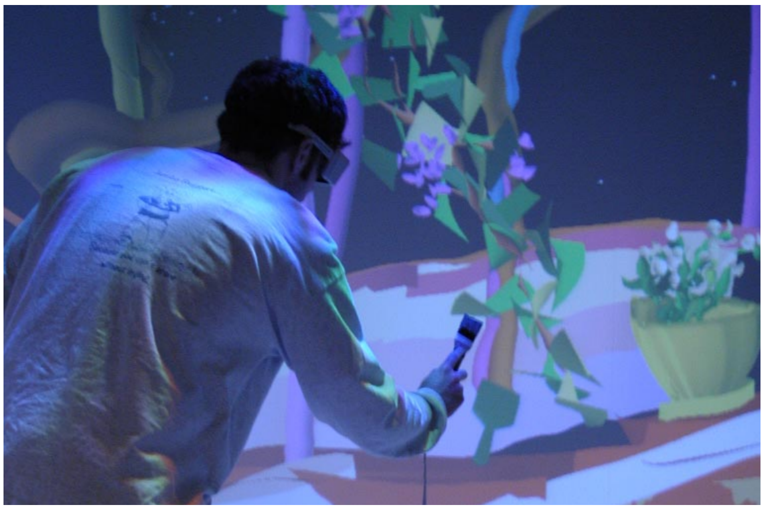


Figure 3.1: *CavePainting* by [Keefe et al., 2001]. Different "brushes" enable painting with volumes or surfaces

The idea of immersive 3D modeling is not novel.

glasses. They discovered a big potential for rapid prototyping, but noticed a need for constraints such as snapping to a grid or drawing projectedly onto planes in order to accommodate for imprecise human movements.
Many authors have picked up on the idea of immersive modeling, but used more abstract modeling techniques. [Schkolne et al., 2001] presented an interface for painting surfaces in AR. As the AR display, they used a monitor and stereoscopic shutter glasses. In their system, the user paints surfaces using hand gestures. By bending their hand, users can manipulate the stroke's cross-section. This modality however is designed for an early exploration phase, rather than creating production designs.

One way to create volumes from line input is to use lines with thickness.

[Keefe et al., 2001] implemented a system for painting in 3D in a cave environment, where different types of strokes were used as 3D paint. Similarly, [Ponto et al., 2013] implemented 3D painting in a cave using volumetric strokes.

Another way to generate volumes from lines is to create curve networks first, and then span faces across them.

FreeDrawer by [Wesche and Seidel, 2001] let the users create complex objects by first sketching them as curve networks and later filling the areas between the curves with surface patches (Figure 3.2). The system by [Jackson and Keefe, 2016] relies on filling curve networks as well, but instead of sketching the curves mid-air, 2D

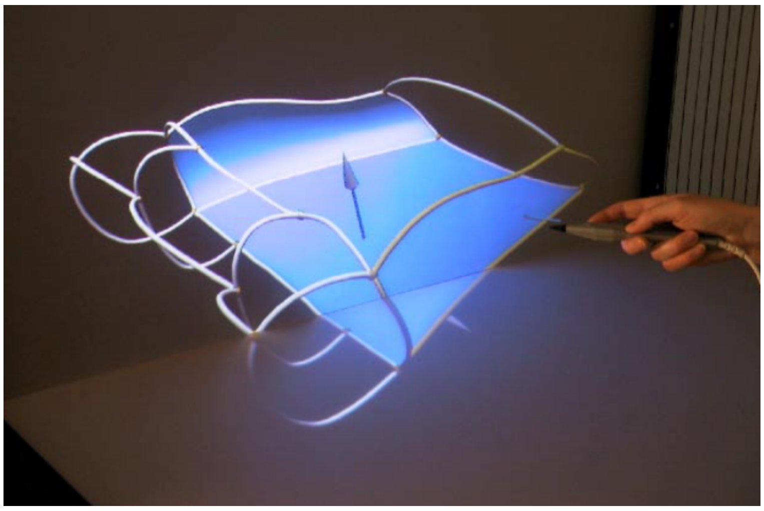


Figure 3.2: WireDraw by [Wesche and Seidel, 2001]. The designer first creates a network of curves, and later spans surfaces through closed loops.

curves are "lifted off" the canvas. This integrates well with initial 2D sketches, and mitigates the issues with mid-air sketching (ref. Section 3.2 "Limitations of Mid-Air Sketching"). [Milosevic et al., 2015] also built a solution based around sketching curve networks, but their system tries to automatically reconstruct the surface between the curves. Due to the bendable pen tip, their system can also be used to 3D scan highly concave objects by tracing the pen along the surface. As the researchers point out, this also makes it very suitable for repair tasks.

Sketching can be used for modeling volumetric objects as it is done with solid modeling software.

Next to systems using pen-like pointing devices as input, there are also systems based on hand gestures. *MockupBuilder* ([Ara, 2012]) used finger gestures combined with bi-manual interaction to enable 3D modeling in a CAD/direct manipulation fashion (Figure 3.3). Furthermore, the system enabled both spatial 3D as well as 2D touch gestures on the surface. As *MockupBuilder* used a CAD-like modeling style, 2D sketching on the surface was used for the definition of profile shapes. Via a shortcut, users could temporarily transform the scene, so that any virtual surface could be mapped onto the table top for sketching on it.

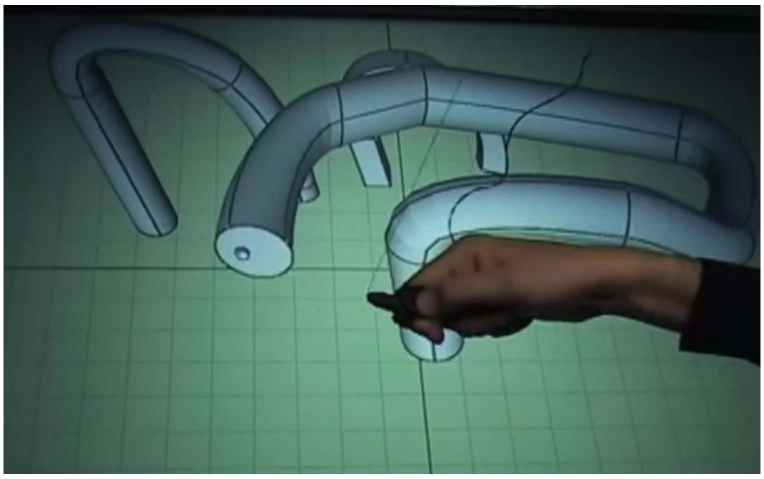


Figure 3.3: *MockupBuilder* by [Ara, 2012]. The designer draws profile shapes on the tablet and sweeps then via mid-air gestures.

Boolean operation are a useful addition.

In *MixFab*, [Weichel et al., 2014] used a plethora of bi-manual hand gestures to create and manipulate 3D geometry in an enclosed AR space. Their system also made use of Boolean operations, which was well received among the users. Also highly appreciated, was the ability to integrate existing objects into the scene by 3D scanning them. Combined with Boolean operations, this enabled e.g. quickly creating a phone stand which perfectly fit the given phone.

Lastly, there are AR modeling tools based on projecting 2D sketches into the 3D scene.
[Xin et al., 2008] used a handheld screen as the AR display. The user could define virtual surfaces (including non-planar ones) inside the scene in various ways using the touch screen and use the touch screen to draw on them projectedly.
[Arora et al., 2018] took this idea one step further by combining the 2D touchscreen input with 3D mid-air input. Users could now define the virtual surfaces to draw upon mid-air, or simply sketch unconstrained in 3D. A Microsoft HoloLens (optical see-through) was used as the AR display.
In *Window-Shaping* ([Huo et al., 2017]), users could create

projected sketches on real-world objects using a tablet PC, and inflate them into volumetric objects using different functions. The created objects lent their texture from the underlying real-world object. The method is therefore very suited for augmenting existing objects for conceptual designs.

3.2 Limitations of Mid-Air Sketching

Mid-air sketching is imprecise compared to constrained sketching in 2D.

Many modeling approaches require the user to create sketches at some point. However, the issues people already have with modeling in 2D are exaggerated in a 3D mid-air scenario ([Arora et al., 2017]). While [Wiese et al., 2010] found that freehand sketching is learnable to a degree, the general issue still remains.

Physical surfaces highly enhance sketching accuracy, but also virtual surfaces can make a slight improvement.

[Arora et al., 2017] further explored the issues and possible solutions for imprecise mid-air sketching and found that e.g. incorporating virtual surfaces as guides into the application can improve accuracy by up to 17%. A higher improvement could only be achieved by directly displaying the target curve (57%). [Wacker et al., 2018] investigated further on the sketching accuracy on physical surfaces compared to sketching on virtual ones and found that having physical guidance improves sketching accuracy significantly, independently from the shape of the physical object.

Smartphone AR incorporates additional potential error sources.

The just mentioned studies have been carried out either in VR ([Arora et al., 2017], [Wiese et al., 2010]) or in AR with optical see-through glasses ([Wacker et al., 2018]). [Kruijff et al., 2010] provide arguments, that the depth perception issues for handheld AR without stereoscopic view (such as with the ARPen) will be even bigger. As reasons for this, they propose e.g. the lack of stereoscopy, a different field of view (FOV) of the device, display properties and viewing angle offset (Figure 3.4). [Čopič Pucihar et al., 2013] made an experiment to quantify the effect related to viewing angle offset and found a significant influence of this factor alone on the targeting accuracy. However, they noticed learning effects in the device-perspective rendering scenario.

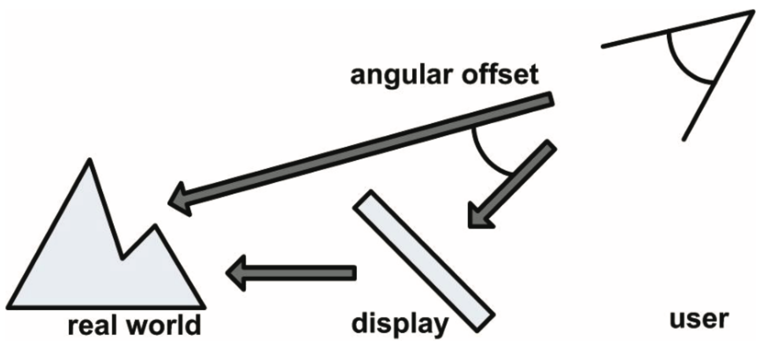


Figure 3.4: Depiction of the viewing angle offset from [Kruijff et al., 2010]

3.3 Operations in CAD Modeling

Based on a set of test case objects by [Li et al., 2010] for which the exact CAD commands used were available, [Kang et al., 2012] aggregated the rates at which they appeared (Table 3.1). Due to the method of [Li et al., 2010], Boolean operations are not included. Instead, there is is a distinction between protruding (i.e. volume generating) and cutting functions.

Extrusions and revolutions are frequently used methods for creating shapes in CAD tools.

When grouping protruding and cutting functions, we see a clear dominance of extrusions (57 times), followed by revolutions (14 times). General sweeps are less common (4 times). Another very common operation was chamfering edges. Aside from the geometry-manipulating functions, it was common to create datum planes and axes, and to repeat objects in linear or circular patterns.

3.4 Solid Modeling Interactions

As just shown, extrusions, revolutions, general sweeps and lofting together make up a large portion of modeling operations used in practice. While technically, extruding, general sweeping and revolving could be a single operation, as already hinted in Section 2.2 "Solid Modeling", the sheer

Feature function	**Frequency of appearance**
Extrusion	33
Cut extrusion	24
Revolve	6
Revolve cut	8
Sweep	1
Sweep cut	3
Loft	2
Shell	2
Linear pattern	9
Circular pattern	6
Fillet	5
Chamfer	11
Datum plane	7
Datum axis	2
Counterbored	1
Countersunk	1

Table 3.1: Appearance frequency of feature functions [Li et al., 2010]

dominance of extrusions and revolutions justifies creating separate operations for them. Also lofting could be though of as a type of sweep, in which the cross-section interpolates between the profiles.

While "everything is a sweep", separate applications-specific operations may be more convenient.

There seems to be a rough consensus between CAD tools on which input techniques to use for these operations: An orthogonal extrusion does not require the explicit definition of a path along which to extrude the profile. The software provides means for defining the extrusion as a magnitude along the normal direction. Only for a general sweep, the definition of a path is necessary (verified for recent versions of Fusion 360[1], SolidWorks[2], CATIA[3], FreeCAD[4]). This way of defining a general sweep is also the approach seen in sketch-based CAD concepts like [Eggli et al., 1997], [Bimber et al., 2000],

[1]https://www.autodesk.com/products/fusion-360/overview/
[2]https://www.solidworks.com/
[3]https://www.3ds.com/products-services/catia/
[4]https://www.freecadweb.org/

[Igarashi and Hughes, 2001],[Pereira et al., 2004] and [Kim and Kim, 2006]. Revolving is usually done by explicitly defining a rotation axis around which to turn the profile. For lofting, multiple consecutive profiles are stacked on top of each other.

Tinkercad is the tool that stands out in this regard. Here, extruded or revolved solids can be added to the scene, but the profile shapes only exist inside a separate editor window as a parameters for the operation at hand. The scene itself can only contain volumetric objects.

Aside from these typical gestures, we found some alternative solutions in literature. For revolutions, [Eggli et al., 1997] decided to let the user sketch the mirror-inverse profile. In [Pereira et al., 2004], the user implicitly defined the revolution by drawing a circle with the desired diameter.[Bimber et al., 2000] and [Kim and Kim, 2006] refrained from using an extra stroke to define the revolution, and simply revolved the sketch around the axis passing through its start and end point. This however makes revolving closed profiles (e.g. to create a ring) impossible.
For sweeps, the gesture space is less diverse. We found no other sketch-based approaches in literature than to define a path along which to sweep the profile. However, based on [Hoffmann, 2017] and other previous observations on people's usage of the ARPen, we concluded, that it would be worthwhile to investigate the possibility to define a sweep by sketching a second profile in the desired target position.

3.5 Summary

In the past there have been numerous attempts at creating immersive 3D modeling systems. However, many of the aforementioned solutions lie more in the VR domain, using either VR glasses, cave setups or setups similar to caves, in the sense that there is a stationary display which displays holographic content when seen through special 3D glasses. With few exceptions ([Milosevic et al., 2015], [Ara, 2012], [Weichel et al., 2014]), these systems were evaluated for purely mid-air interaction, not taking into account the ad-

Not many previous solutions were conceptualized/evaluated with in-situ modeling in mind.

vantages of physical surfaces as guidance. However, the approach by [Milosevic et al., 2015] was very different, in the sense that it used the sketches only as a scaffold around which the actual geometry was automatically generated. Also, the interaction in [Ara, 2012] and [Weichel et al., 2014] was heavily based on bi-manual hand gestures, which is not possible with the ARPen. For our study, we therefore wanted to put a higher focus on sketching. Aligning with the findings of [Kruijff et al., 2010] and [Arora et al., 2017], our preliminary experience with the ARPen has shown that users often wrongly estimate the pen tip's position in depth (i.e. the pointing direction of the phone's camera). This often lead to highly disconnected sketches, which reveal only after the user shifts the view.

We hope that having to draw less will reduce error potential.

As the described solid modeling gestures require relatively few input strokes, we suspected them to be less prone to such errors and we generally considered their efficiency to be an advantage. From the projects based on solid modeling gestures as mentioned in Section 3.4 "Solid Modeling Interactions", only [Bimber et al., 2000] made use of physical guidance, but only regarding drawing on a handheld tablet-like surface.

Inferring from these findings, we therefore decided to use solid modeling techniques as a means to explore 3D in-situ modeling with the ARPen. Furthermore, CSG—which is closely related to the CAD/solid modeling domain—sound be a powerful addition and make sense in an in-situ scenario regarding its use for object fitting ([Weichel et al., 2014]).

Chapter 4

Implementation

This chapter is about the implementation of the core modeling functionality. The implementation of the individual interaction techniques will be subject of Chapter 5 "Modeling Techniques".
We start will start with the elicitation of the functional and non-functional requirements. Then, we will reason about our choice for the library which provides the backend for the geometry calculations. Lastly, we will describe the software architecture of our implementation and briefly illustrate how the modeling functionality is linked to the UI.

4.1 Requirements

We wanted to extend the existing ARPen app s.t. users are be able to:

a.1 Create mid-air sketches that can e.g. act as profiles for general sweeps.

a.2 Create volumetric geometry using general sweeping, revolving and lofting.

a.3 Combine volumetric models using Boolean operations.

a.4 Export models to STL files.

While sketching freehand lines with the ARPen was already possible before, our requirement **a.1** states, that they can be used as an input for further sweeping operations. Therefore, continuous strokes must be recognized as such and should be free of jittering in order to avoid self-intersections when projected onto a 2D plane. This was not the case with the existing implementation. **A.4** had the concrete purpose of exporting models from the user study for later analysis. It is however also useful in general. Technically, this requirement was already met by SceneKit, but an stereolithography (file format) (STL) export functionality on a lower level would allow exporting at a much larger detail than the preview versions which are shown while editing.

The non-functional requirements were:

b.1 All calculations are performed on the device hardware.

b.2 All included libraries are free and open-source.

b.3 There is immediate visual feedback for all user inputs.

Requirements **b.1** and **b.2** ensure the compliance with the concept of the ARPen project, which include making the software easily accessible to the general public. **B.3** was just one way of ensuring that basic usability standards are met, enabling user study participants to judge the essential interactions without being distracted by avoidable usability issues.

4.2 Modeling Kernel

CAD libraries cover large portions of our requirements.

For performing the geometrical calculations we used Open CASCADE Technologies (OCCT) as the geometric modeling kernel. The reason for choosing a library from the CAD domain was that they all fundamentally provide solid modeling functions, including CSG (requirements **a.2**, **a.3**). The biggest distinguishing characteristic of the libraries we considered (Table 3.1) was the data structure by which they represent the geometry internally. The major modeling kernels *ACIS*, *C3D*, *Open Cascade* and *sgCore* (Table 4.1) use B-Rep. This has the advantage, that the models can have arbitrary detail down to floating-point accuracy. For rendering or STL export, the models can be converted into a mesh at the desired granularity. The downside of this is, that the triangulation has to be performed every time a change in the model should be shown to the user. Another trade-off with B-Rep is that the surfaces created via extrusions, revolutions, etc. are—at least in *OCCT*—not further deformable beyond linear transformations. It would therefore be hard to e.g. add digital sculpting capabilities to our system in the future[1].

B-Rep is current the de-facto industry standard, but may limit flexibility.

These issues do not occur with mesh-based libraries such as *Euclid* and *CGAL* (popular for being used in OpenSCAD[2]). Here however, great care has to be taken about the level of detail by which to create the primitives early on, as it is only partly adjustable later[3]. A way to circumvent this issue is to use a completely parametric modeling approach as done in *OpenSCAD*.
One library we looked at, *libfive*, uses F-Rep. This enables unique operations such as blending between objects.

Unfortunately, we could not use *sgCore*, *C3D* and *ACIS* in the *ARPen* project due to their proprietary licenses. Furthermore, we could not find evidence that *CGAL* and *libfive* can be compiled for iOS, leaving only *OCCT* and *Euclid*.
Euclid is written in Swift and explicitly compatible with SceneKit. It can therefore be used to apply Boolean operations to SceneKit meshes[4] without the need for a conversion to B-Rep, at the cost of having continuous and numerically precise geometry. We decided to use *OCCT* nevertheless because of the following reasons:

Many libraries were excluded due to incompatibility with the ARPen project.

- Being published on GitHub in December 2018[5], *Euclid* was very new at the time, raising concerns about its stability and future maintenance. *OCCT* on the

[1]It would be possible on the generated meshes, but this would mean sacrificing the numerical precision and the non-destructive workflow
[2]https://www.openscad.org/
[3]e.g. via subdivision or decimation.
[4]There might be restrictions regarding watertightness.
[5]https://github.com/nicklockwood/Euclid/

Library	Data struct.	Bool.	Sweep	Extrude, Revolve, Loft	Shell	iOS	License
OCCT[a]	B-Rep	✓	✓	✓	✓	✓	LGPL 2.1
Euclid[b]	Mesh	✓	✗	✓	✗	✓	MIT
sgCore[c]	B-Rep	✓	✓	✓	✓	✓	Propr.
libfive[d]	F-Rep	✓	✗	✓	✗	?	LGPL 2.1
CGAL[e]	Mesh	✓	✓	✓	✓	✗	LGPL v3+
C3D[f]	B-Rep	✓	✓	✓	✓	?	Propr.
ACIS[g]	B-Rep	✓	✓	✓	✓	?	Propr.

Table 4.1: Comparison of selected CAD kernels.

[a]https://www.opencascade.com/
[b]https://github.com/nicklockwood/Euclid/
[c]http://www.geometros.com/
[d]https://libfive.com/
[e]https://www.cgal.org/
[f]https://c3dlabs.com/
[g]https://www.spatial.com/

other hand is a very mature framework, first released in 1993[6] and used in a plethora of projects.

- For the time being, *Euclid* did not feature general sweeps and shelling, which we considered to be important operations.

- We considered modeling with arbitrary detail to be desirable in the personal fabrication domain.

As building *OCCT* for the ARPen project turned out to be a non-trivial task, we provide a detailed documentation (see file below) of our method, in case there will be the need to replicate the process in the future.

File: compileOcct[a]

[a]http://hci.rwth-aachen.de/public/ARPen/compileOcct.html

[6]https://www.opencascade.com/content/company

4.3 Architecture

Our implementation is structured as a basic *protocol-based layer architecture* ([Zllighoven, 2004]). Three layers provide increasing abstraction of the low-level calls to OCCT (Figure 4.1). In this section, we will explain one layer at a time, going from the lowest to the highest one.

Geometry Processing Layer

The geometry processing layer accesses OCCT.

The geometry processing layer is where the manipulations to the geometry are being performed. It is the only part of the software where calls to *OCCT* are made. Therefore, it encapsulates *OCCT* from the rest of the codebase. It is written in Objective-C++, but as the calls to *OCCT* have to be done in C++ syntax, we decided to use C++ consequently for a more consistent style. Only the function headers are written in Objective-C in order to be exposable to Swift via the bridging header. For clarity, the functionality is split into the following files:

Registry.mm Responsible for memory management.

Builders.mm Contains the geometry instantiation- and manipulation logic.

Helpers.mm Provides helper functions like math utilities.

Meshing.mm B-Rep to mesh conversion for SceneKit and STL export

Swift API Layer

The Swift API layer provided Swift-like access to the geometry manipulation layer.

As many data types commonly used in Swift are different from the ones used in Objective-C/C++, we introduced another abstraction layer for convenience. The API is similar to the one exposed by the Objective-C headers, but uses Swift data types. Conversion between the data types is done here, as well as error handling.

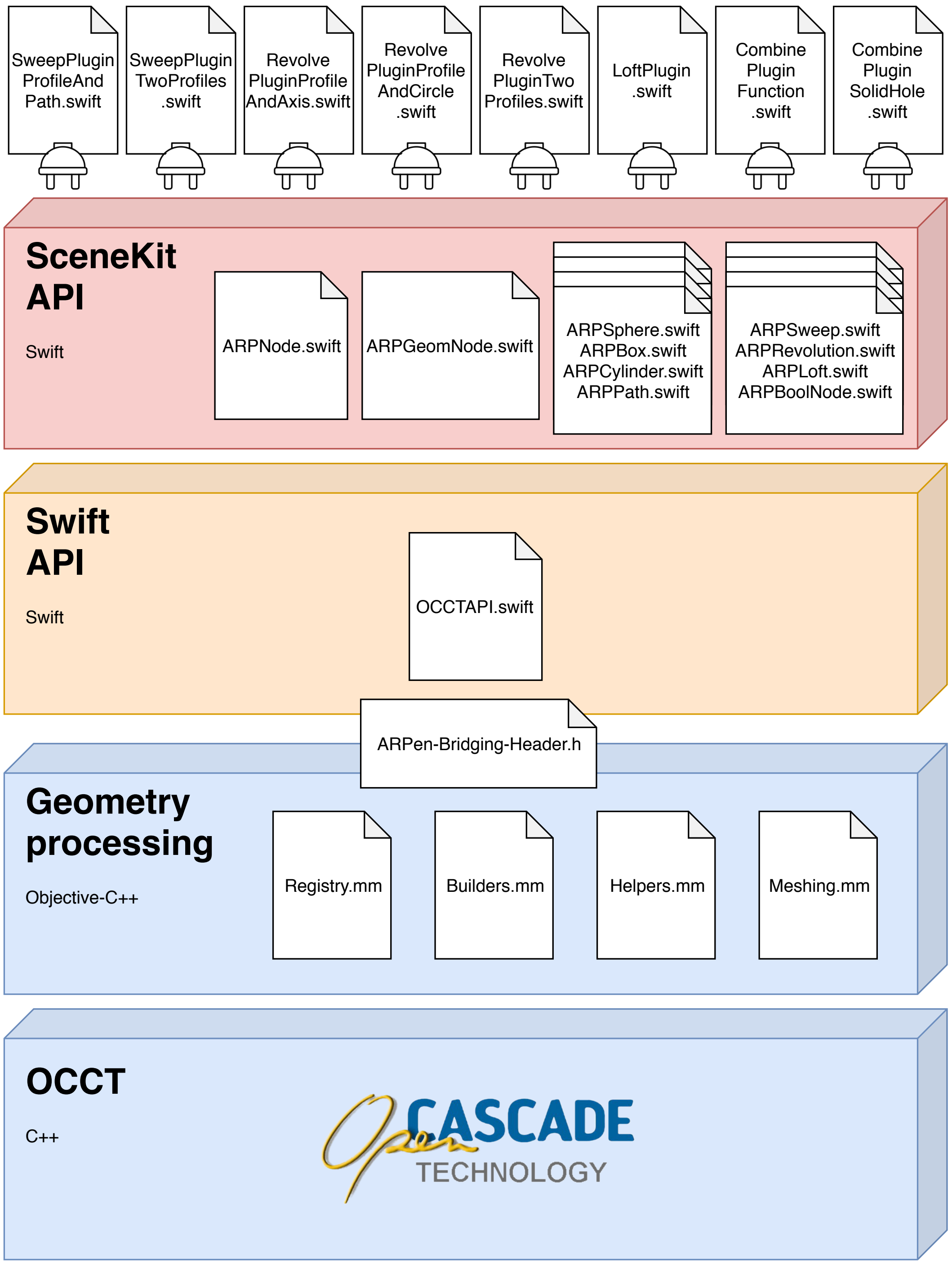

SweepPlugin ProfileAnd Path.swift
SweepPlugin TwoProfiles .swift
Revolve PluginProfile AndAxis.swift
Revolve PluginProfile AndCircle .swift
Revolve PluginTwo Profiles.swift
LoftPlugin .swift
Combine Plugin Function .swift
Combine Plugin SolidHole .swift
SceneKit API
Swift
ARPNode.swift
ARPGeomNode.swift
ARPSphere.swift ARPBox.swift ARPCylinder.swift ARPPath.swift
ARPSweep.swift ARPRevolution.swift ARPLoft.swift ARPBoolNode.swift
Swift API
Swift
OCCTAPI.swift
ARPen-Bridging-Header.h
Geometry processing
Objective-C++
Registry.mm
Builders.mm
Helpers.mm
Meshing.mm
OCCT
C++
Open CASCADE TECHNOLOGY

Figure 4.1: System architecture divided into layers.

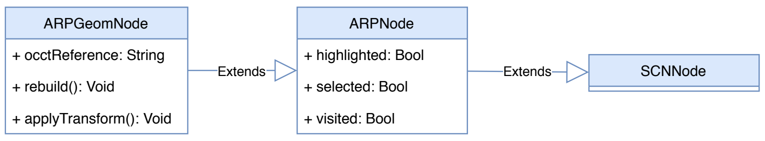


Figure 4.2: Inheritance hierarchy of the fundamental geometry class.

SceneKit API Layer

The SceneKit API layer couples SceneKit with OCCT.

This layer provides the interoperability between *OCCT* and SceneKit.
`ARPGeomNode` is the base class for most other classes in this layer. It extends `SCNNode`, adding the functionality to synchronize the node with a `TopoDS_Shape`[7] (*OCCT*'s geometry representation) in the geometry processing layer. Its inheritance hierarchy and basic API are shown in figure 4.2. A string identifier is used as the key to reference the underlying geometry. Each subclass of `ARPGeomNode` has to override the function `build()`, providing the logic on how to generate the shape.
Calling the function `rebuild()` forces the shape to be built, triangulated, converted to a SceneKit mesh and appended to the node. The function `applyTransform()` applies the current transform of the node to the underlying representation and has to be called manually.

We implemented a hierarchy of operations.

Figure 4.3 shows three simple geometry classes and `ARPBoolNode`, which can be used to combine two `ARPGeomNode`s using a Boolean operation. This demonstrates the hierarchical nature of our approach: As en example, two nodes `a` and `b` are passed to a new `ARPBoolNode` `c` in the constructor and automatically become its children. Changes can be applied to `a` or `b` afterwards, resulting in `c` getting rebuilt. These nesting hierarchies may be arbitrarily deep.

The `ARPPath` class lays the foundation for creating sketches (Figure 4.4). A path consists of a finite amount of

[7] `https://www.opencascade.com/doc/occt-7.3.0/refman/html/class_topo_d_s___shape.html`

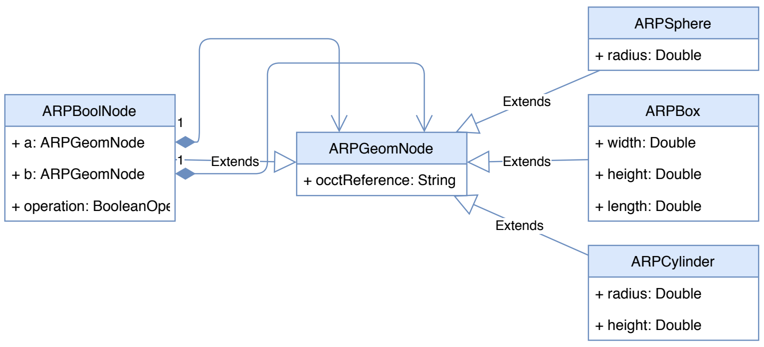


Figure 4.3: Primitive shape generators and their boolean combination.

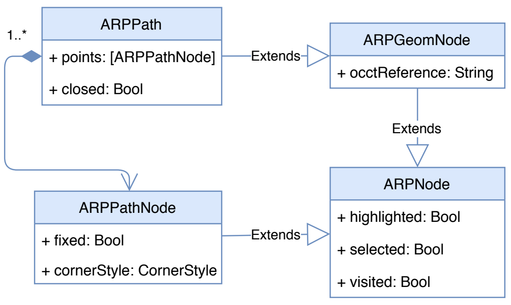


Figure 4.4: Composition of a path object.

Sketches are defined as multipoint paths with round or sharp corners.

`ARPPathNode`s and can be open or closed. Each node's `cornerStyle` can either be *sharp* or *round*, whereby a B-spline is fitted through any sequence of *round* nodes, while no continuity constraints are imposed on *sharp* nodes. There are shapes which can not be expressed this way (e.g. a smooth transition from a straight segment to a round segment) but we decided to not further extend the spline creation capabilities in order to reduce complexity.
The creation of the path based on the nodes is done on the geometry processing layer. A mesh is created from the generated path by creating low-poly cylinders from path segments sampled in regular intervals. The mesh is then passed back to the SceneKit implementation for display, leaving the SceneKit API layer essentially unaware of the path's actual shape.

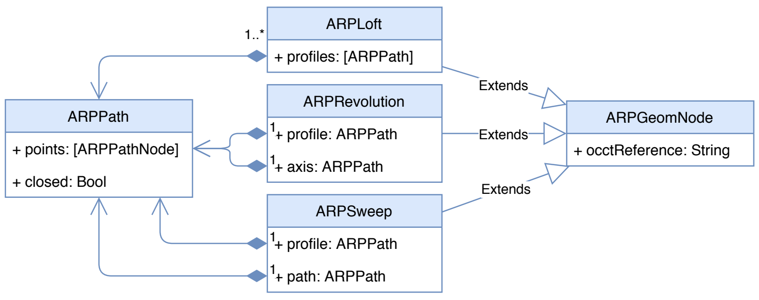


Figure 4.5: Composition of sweeping operations.

Just as an `ARPBoolNode` can be used to combine two `ARPGeomNode`s, `ARPLoft`, `ARPRevolution` and `ARPSweep` can be used to combine paths in order to generate volumes using the respective operation. Also here, our hierarchical approach allows the shape to be modified retrospectively by altering the input paths.

4.4 Plugins and UI

For separating the different functionalities, we used [Wehnert, 2018]'s plugin system. As shown in Figure 4.1, each every function under every technique is contained in a separate plugin. The interaction is mode-based, e.g. while having one of the sweep plugin selected, the sweeping action is triggered as soon as the necessary paths are found in the scene. A screenshot of the UI can be seen in Figure 4.6.

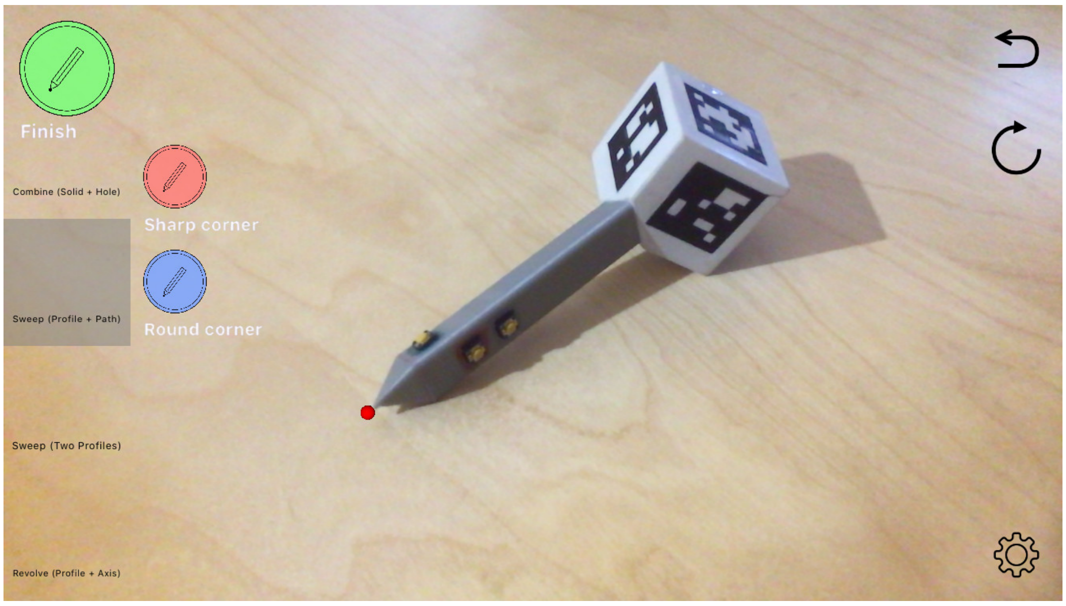


Figure 4.6: Screenshot of the app. The menu on the left is used to select the active plugin. The green, red and blue software buttons map to the respective hardware buttons on the pen and can be used if the hardware buttons are not available. The text underneath indicates the current plugin-dependent function of each button.

Chapter 5

Modeling Techniques

In this chapter, we describe the concept behind the modeling functions and their respective techniques, which were evaluated in the user study. We begin by explaining how sketching and selection/translation—the basic functions on which the others build upon—are realized. We then describe all sweeping-related techniques and the Boolean operations.

5.1 Sketching

As we chose our interactions to be derived from typical solid modeling operations, creating sketches is a fundamental element on top of which most other methods build upon. Unfortunately, previous experience with the ARPen has shown, that the tracking is currently not reliable enough in order to use its raw input for line drawing. If a stroke contains self-intersections due to jittering, this may render it unsuitable to be used as an extrusion profile. One option to circumvent this would have been to implement post-processing of the strokes, but we reasoned that the expected utility would not justify the implementation effort. Instead, the sketching in our implementation is based on creating polylines with an option to create curved

We chose to implement the creation of sketches through multipoint lines, as it was an easy and reliable way of dealing with jittery input.

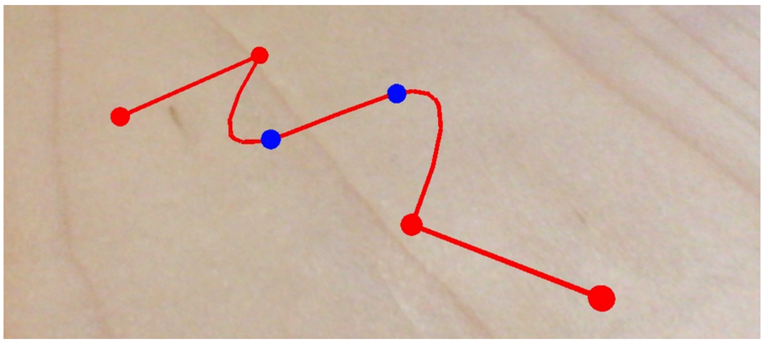


Figure 5.1: A path created with the ARPen. Red points are sharp, blue points are round corners.

segments using B-Splines[1], which are fitted through user-defined control points (Figure 5.1). When holding down a button, control points are created in distance-wise regular intervals, simulating freeform sketching.

5.2 Selecting and Translating

For selecting and translating objects, we implemented techniques similar to *pen ray* and *pen ray pickup* from [Wacker et al., 2019]: An object is in focus if it is hit by a ray cast from the device camera through the pen tip. Selection/deselection is done via a button press. When the pen is moved by a threshold while holding the button, the selected object snaps to the pen tip an then moves along with it until the button is released.

It is possible to traverse the hierarchy of objects and thereby alter them retrospectively.

Via a double-click, objects can be "visited", meaning that their children (if applicable) get visible and editable again. This way, operands of a Boolean operation can be rearranged and paths can be edited.

[1]https://www.opencascade.com/doc/occt-7.3.0/refman/html/class_geom___b_spline_curve.html

5.3 Sweeping

In the following, we will describe the sweeping techniques we decided to incorporate into our solution in order to be evaluated and compared in the user study.

General Sweeps

As shown in Section 3.3 "Operations in CAD Modeling", extrusions are the most common type of sweeping functions in CAD modeling. However, they only represent a special case of general sweeping. In an attempt to reduce the necessary amount of functions in our system—and thereby reduce the complexity—we therefore decided to unify them in our solution. Thus, an extrusion is simply a general sweep along the normal direction. We will reflect upon this decision in Section 6.8 "Discussion". In the following, we will present the two techniques we decided to implement and evaluate.

Extrusion is just a special case of sweeping in our system.

Profile and Path

Sweep (Profile + Path) (SPP) is inspired by the traditional way of creating swept volumes in CAD software. The user specifies a profile (which has to be closed in our case, as we only want to generate volumes) and a path along which to sweep the profile (Figure 5.2 left). When the plugin is active, the system automatically executes the command as soon as both paths are specified. Our method only handles planar profiles, which is why they are "flattened" using singular value decomposition (SVD).
The path does not have to start exactly on the profile. In fact, it could start anywhere in the scene. While being swept, the profile maintains the relative orientation it had to the path at its start. At discontinuous locations, the profile does not change its orientation.

SPP is the "traditional approach".

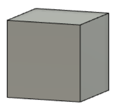
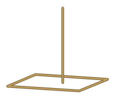
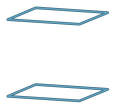
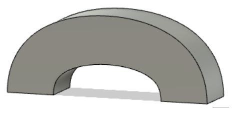
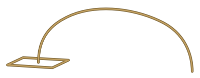
	Profile and Path	Two Profiles
		
		

Figure 5.2: Two example models and how to create them via SPP and S2P.

Two Profiles

S2P is an alternative input method inspired by our preliminary observations of people using the ARPen. The user defines the profiles at both ends of the shape and the system interpolates between them. Contrary to lofting, the profiles do not have to lie on top of each other. We designed it in way, that up to a certain threshold of offset between the profiles, the system would create a straight connection. Otherwise it would create are curvy connection which aims to minimize bending (a real-world analogy would be a Slinky[2]) (Figure 5.2 right).

S2P works by connecting two profile shapes.

Partly for practical reasons, we decided to only use the second profile for specifying the target position and orientation of the sweep. The start profile is the one that defines the cross-section at every point. Therefore, the actual shape of the second profile is not important, as long as it defines a plane (i.e. contains at least three nodes). To calculate its target position, we average the positions of the end profile's nodes.

Revolving

Our implementation of revolutions differs from typical CAD software in the way, that revolving an open profile will not create a surface without volume. Instead, the sys-

[2]https://en.wikipedia.org/wiki/Slinky

tem will create a metaphorical "lid" and a "base" for otherwise open objects. Creating hollow objects such as cups or vases is therefore currently hard to achieve solely by revolving a profile, but can be done in at a later stage using Boolean operations. Also, techniques such as shelling could be added later on.

Profile and Axis

Revolve (Profile + Axis) (RPA) is the "traditional approach".

Like SPP, Revolve (Profile + Axis) (RPA) is the technique which is common in CAD software, the only difference being the creation of volumes as described above. After specifying the profile, the user defines the revolution axis as a simple path between two nodes (Figure 5.3 left). We see the advantage of this method in its straightforwardness and the minimal input required.

Profile and Circle

Strictly speaking, every revolution could by created using a general sweep operation as well, which is why they are often also referred to as "rotational sweeps". Revolve (Profile + Circle) (RPC) can be seen as an implementation of this concept. In RPC the user defines rotation implicitly by sketching a circle (Figure 5.3 center). In order to derive a rotation axis from it, the sketched circle path is flattened using SVD and a *linear least squares fitting circle* ([Coope, 1993]) is fitted through its nodes. The rotation axis is then the line which is orthogonal to the circle and passing through its center. While this may mean, that the circle drawn by the user will not perfectly map to the final revolution being executed, we hope that it will help the user by smoothing out errors.

RPC reflects a revolution's nature of also being a sweep along a circular trajectory.

Two Profiles

Inspired by [Eggli et al., 1997], in R2P the user completes the revolution by sketching the diametrical version of the

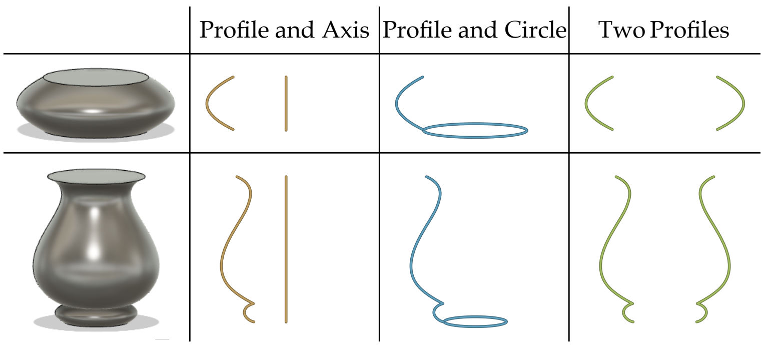


Figure 5.3: Two example models and how to create them via RPA, RPC and R2P.

R2P may facilitate the estimation of the diameter.

profile. The revolution axis is determined by fitting a line through the coordinates exactly between the first- and last points of the original- and the mirrored profile (Figure 5.3 right). As with S2P, the second profile is not actually used aside from helping define the revolution axis, and might therefore just as well be approximated by a simple line. We hypothesize, that just as with RPC, this method will make it easier to correctly estimate the width of the object.

Lofting

In lofting mode, the user can draw an arbitrary number of closed profiles at increasing heights, which will be connected to create a closed volume. This is identical to how it is depicted in Figure 2.2.

5.4 Boolean Operations

Union and *difference* are the two supported Boolean operations.

Boolean operations can be used to create complex shapes by combining geometry. While CAD software often supports the logical operators *union*, *difference* and *intersection*, we decided to follow the example of Tinkercad[3] and only implement *union* and *difference* for simplicity.

[3]https://www.tinkercad.com/

Function

In *function* mode, the user selects two objects and executes either *merge* or *cut* (synonyms for *union* and *difference*). When merging, the two objects will be combined into a single one. When cutting, the secondly selected object will be cut away from the first (Figure 2.2).

Solid/Hole

As inspired by Tinkercad, the user can toggle an object between being a *solid* or a *hole*. When two *solids* are combined, it is the equivalent of executing the *merge* command. When a *solid* and a *hole* are combined, the *hole* is cut away from the *solid*.

Chapter 6

User Study

The goal of our study was to observe people working with our sketch-based 3D modeling system and gather feedback. The purpose of the different techniques was to provide the participants with a set of functioning alternatives, in order to give them a basis for argumentation. Simultaneously, we were able to use quantitative metrics to draw first inferences about concrete differences in task performance between the techniques. In appropriate places, null hypothesis significance testing is used in order to support our findings.

The original idea of the user study was to evaluate all working features: General sweeps, revolutions, lofting and Boolean operations. However, as it turned out during the pilot studies, evaluating only the general sweeps and revolutions was already hard to fit into a 60 minute study session. Due to their conceptual similarity, we therefore decided to only evaluate the sweeping-related methods, leaving the evaluation of lofting and Boolean operations for future work.

Only general sweeps and revolutions were evaluated due to time constraints.

In the following, the term "sweep" will be used as a shorthand to describe general sweeps.

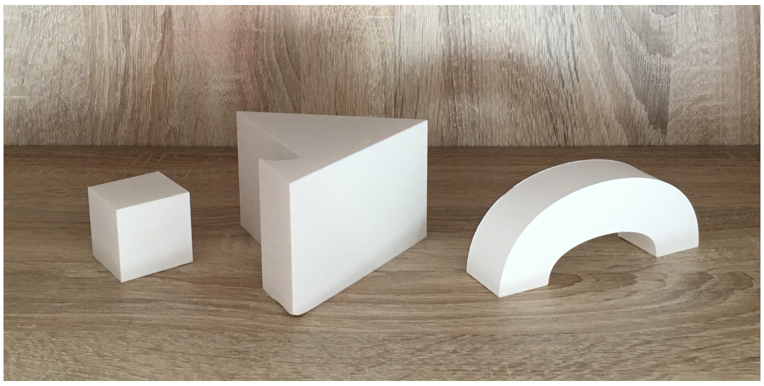


Figure 6.1: The reference objects for the sweeping study: *Cube* (left), *Phone stand* (center) and *Handle* (right).

6.1 Experimental Design

The task of the user study was to recreate a set of objects.

In order to enable the participants to compare the techniques, we designed the study as a within-groups design. Using the 5 techniques (2 for sweeping and 3 for revolving) described in Chapter 5.3 "Sweeping", the participants had to recreate a set of 3D-printed reference objects which are going to be explained in the following. All models would fit inside a cube with an edge length of 11cm. We asked the participants to do 3 repetitions for each combination of *model* × *technique*, so that they could familiarize themselves better, and we would have the chance to search for eventual learning/correction effects. In order to avoid practice effects and progressive error due to the within-groups design, we counterbalanced the order of *model* × *technique* using a Latin square for both the sweeping and revolving tasks. Thus, the study required a multiple of 6 participants.

Sweeping

The models to be recreated by a sweeping operation are depicted in Figure 6.1.

Cube The *Cube* (Figure 6.1 left) was designed as an object with both a simple profile and a simple path.

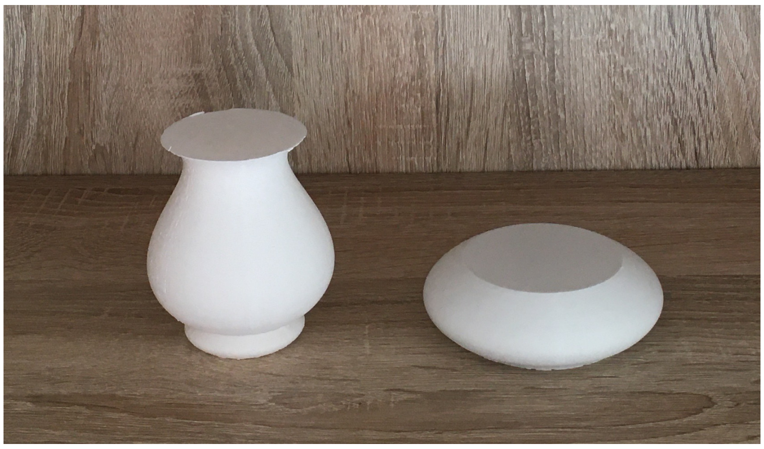


Figure 6.2: The reference objects for the revolution study: *Flower pot* (left) and *Door stopper* (right).

Phone Stand The *Phone stand* (Figure 6.1 center) is also simply extruded in normal direction but has a more complicated profile.

Handle The *Handle* (Figure 6.1 right) has a simple profile, but it is swept along a more complex trajectory. We made sure, that S2P would yield this exact shape if the profiles are drawn correctly.

This results in the number of trials:
2 (techniques) $\times$ 3 (models) $\times$ 3 (repetitions) = 18 (trials per participant)

Revolving

The models to be recreated by a revolving operation are depicted in Figure 6.2.

Door Stopper The *Handle* (Figure 6.2 left) has a simple profile, in the sense that it has few nodes and the first and last ones are above each other. In case of R2P, this means that no skewness has to be taken into account when approximating the mirrored profile by a line. We however designed it to have a comparatively large radius, expecting this to complicate the correct size estimation.

Flower Pot The *Flower pot* (Figure 6.2 right) has a more complex profile. We suspected it to be more difficult for R2P, for the reason just mentioned.

This results in the number of trials:
3 (techniques) $\times$ 2 (models) $\times$ 3 (repetitions) = 18 (trials per participant)

6.2 Task

To speed up the study, the profile shapes were given.

Each task was to recreate the given reference object as closely as possible using the prescribed technique. In order to save time, the respective profile shapes already existed in the scene, meaning that it was the participant's task to complete the volume-generating operations.
After each trial, the participant was allowed to assess the quality of their created model, including moving the reference object next to it to compare them side by side.

6.3 Quantitative Measurements

We identified multiple quantitative variables to yield potentially interesting insights:

Task completion time The app recorded the task completion times in seconds. The recording started when the first path node was placed, and stopped on finalizing the last path which was necessary for generating the object.

Model quality We measured *model quality* inversely, as the deviation from the ideal result. Different metrics were used depending on the task, which will be explained in the next section.

Reported ease We assessed the perceived ease of each technique in conjunction with each model on a 7 point Likert scale, following the SEQ ([Sauro and Dumas, 2009]) scheme.

Personal preference Participants were asked to rank the techniques by their general preference, independently from the model.

Viewpoints switches We counted how often the participant would switch between viewpoints during task execution. e.g. *Default view* →*Top view* →*Default view* would count as 2.

Second profile For S2P and R2P, we captured how the user specified the second profile. We either denoted the shape, or used *Copy* and *Approximation* to tell that they tried to imitate the given profile perfectly or approximately.

Wilcoxon rank-sum and Kruskal-Wallis tests were used.

Task completion time, *reported ease* and the *model quality* metrics were compared using the Wilcoxon rank-sum test in case of the sweeping techniques (2 levels). For the revolution techniques (3 levels), we used the Kruskal-Wallis test and Wilcoxon rank-sum tests with Bonferroni correction for pairwise post-hoc comparison.

Measuring Model Quality

We invented different quality metrics, but also logged the created objects as STL.

We thought of different way of assessing the model quality, i.e. the similarity of the created model to the original. From the techniques based on comparing the actual 3D meshes we found [Chen et al., 2003] to be the most appropriate in our scenario. However, such similarity values tell little about the cause of the deviation. We therefore decided to use custom quality measures based on our context, which are explained in the following. Nevertheless, we also exported all models created during the trials in STL format for later review.

Sweeps

For sweeps, we defined the *deviation* of a swept model as the positional offset of the end profile's center $\vec{e_p}$ to the center of the end profile in the reference model $\vec{e_r}$, normalized by the distance between the start $\vec{s_r}$ and end $\vec{e_r}$ profile's center of the reference model:

$$d = \frac{||\vec{e_r} - \vec{e_p}||_2}{||\vec{s_r} - \vec{e_r}||_2}$$

For sweeps, we measured how far the end profile was off.

Note, that therefore in case of the *Handle* model, the sweep path was not taken into account by this metric. In general, the skewness of the end profile is not captured as well.

Revolutions

For revolutions, we measured two values: The *radius deviation* and the skewness/*angle deviation*.
We defined the radius of an object as the average of the radius at the bottom r_p^b and at the top end r_p^t of the shape. The *radius deviation* is the absolute difference between the radius of the reference- and the radius of the created model is then normalized by the radius of the reference model:

$$dr = \frac{|r_p^t + r_p^b - r_r^t + r_r^b|}{r_r^t + r_r^b}$$

The *angle deviation* is simply calculated as the angle between the rotation axis' direction $\vec{d_p}$ in degrees relative to the y-axis:

For revolutions, we measured the deviation in size and the skewness.

$$da = \arccos(|y(\vec{d_p})|) * \frac{180}{\pi}$$

Qualitative measures

On top of the qualitative measures, we also gathered the comments made by the participants as well as notable behaviors. To further encourage the participants to speak their mind, we prepared a set of questions, asked at selected points during the study (Appendix D "User Study Questions").

6.4 Apparatus and Setup

We did the study on an iPhone 6s.

The study was conducted using an iPhone 6s. It has a 4.7-inch display at a resolution of 1334 $\times$ 750 pixels and weights 143 grams. To ensure that it wouldn't run out of battery during the study, it had to be constantly connected to a power supply. We made sure, that the cable was long enough to not limit the participant's range of motion. Only one participant made a remark regarding a disturbance by the cable. Note however, that the plugged-in cable provided alternative grips for holding the phone, which may have influenced the physical comfort (positively or negatively).
We used a small web app to control the current task and other settings remotely from a laptop, in order to avoid frequent interruptions during the trials.
To avoid distractions, the study was conducted in a closed room. The participant was seated at a table in front of the wall (Figure 6.3). The proximity of the wall was important in order to prevent the iPhone camera from switching its focus to distant objects, thereby disturbing the optical tracking of the pen's markers. To furthermore facilitate the pen tracking, it was made sure that the room was well lit, but the lighting was soft in order to prevent reflections on the marker cube. The smooth lighting was also meant to facilitate ARKit's world tracking, which we tried to further stabilize by providing many visual cues in the form of feature-rich printed images placed in the scene. One image (Figure 6.3: Image with stones at the back of the table) additionally functioned as a visual marker and was used to ensure that the virtual scene has its origin at a consistent location (right on the table, 20 cm in front of the image).

We aimed for an ideal environment for the ARPen to operate in.

During the trials, the 3D printed reference objects were placed close to the far left corner of the table. They were scaled the same way as the prepared virtual profiles, such that after a flawless trial, it would have been possible to hide the physical object perfectly behind the virtual one from all camera angles.
The trials were recorded using a camera on a tripod, filming from the angle from which the photo in Figure 6.3 was taken.

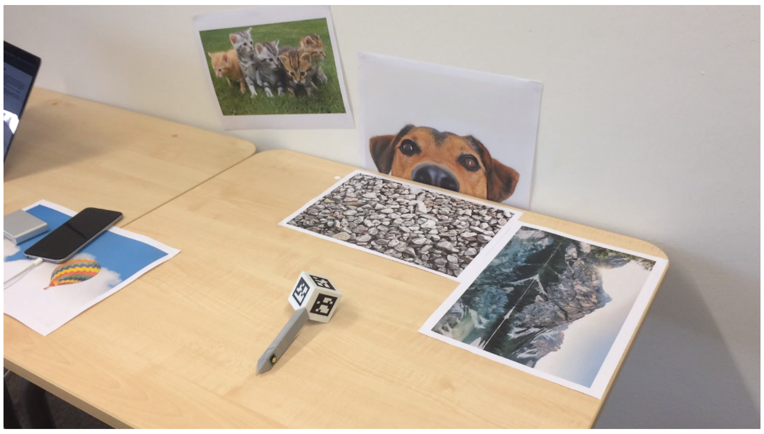


Figure 6.3: The setup of the user study.

6.5 Study Procedure

Each session started by welcoming the participant and educating her about the background and purpose of the study, the rough procedure and the ways in which the data would be collected. After she signed the declaration of compliance, the participant had to fill out the *Participant Information* questionnaire (Appendix A).

In the beginning, there was a tutorial and free modeling session.

The study began with a short tutorial on how to use the pen do draw curves, followed by a free modeling session. The tutorial contained information on the general operation of the system (how to hold the phone and pen, how the pen is tracked) and the ways of drawing paths (the button's functions, how to create sharp and round corners, how to finish a path, how to create a closed path). The participant was told to think aloud during the course of the study.
Once the participant claimed to feel confident, the first part of the study—sweeping techniques—began. For each technique, the experimenter gave a short explanation, followed by a training session until the participant claimed to feel confident. She then had to recreate the objects three times each using the prescribed technique, resetting the scene manually each time by pressing the *Reset* UI button (Figure 4.6 top right). After being finished with all models, she was asked to assess the ease of creating each model on the

After learning a new technique, there was always another training session.

SEQ scale on our laptop. The study then continued with the next technique analogously.
After the sweeping techniques were done, the experimenter asked the *Post Techniques* questions (Appendix D).

The procedure for the second part of the study—revolution techniques—was done analogously.
After both parts were finished, the experimenter concluded the study by asking the *Post Study* questions (Appendix D). The entire procedure took around 60 minutes per participant.

6.6 Participants

We evaluated data from 12 participants.

We conducted the user study with 14 participants, from which two were not included in the evaluation due to incomplete video recordings. The 12 remaining participants were 20 to 27 years old (mean: 23.58 years, sd: 1.83 years). 10 identified as male and 2 as female. 2 of the participants were left-handed, but we found no hints for a systematic deviance in their task performance. The amount of 3D modeling tools that each participant had experience with reached from 0 to 6 (mean: 2.25, sd: 1.71). A list of the tools along with their respective number of users is shown in Figure 6.4. 7 participants had already worked with the ARPen prior to the study.

6.7 Results

In the following, the results of the user study will be discussed. We will first present the results directly related to the sweeping and the revolution study, followed by the general findings which were independent from the techniques (6.7). Finally, we will go through our findings regarding the ARPen system in general.

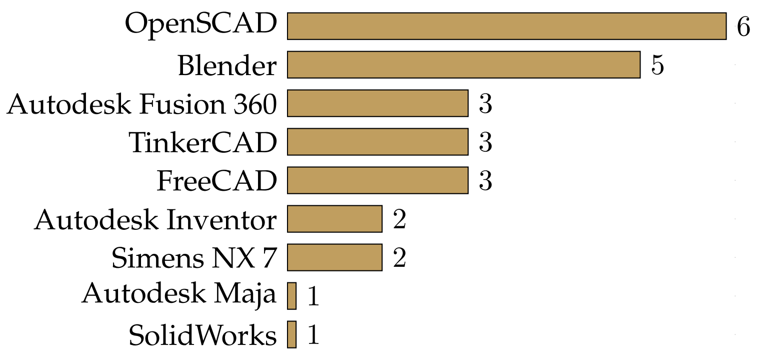


Figure 6.4: Number of participants who had experience using the respective 3D modeling software.

Sweep

As expected, the results of the sweeping study varied across the models, making it impossible to determine a universally superior technique. We noticed a general tendency, that SPP worked better with simple perpendicular sweeps (*Cube*, *Phone stand*), while S2P was superior for curved sweeps (*Handle*). In the following, we will report our findings based on this distinction.

Perpendicular Sweeps

SPP worked better with perpendicular sweeps.

For simple, perpendicular sweeps (*Cube*, *Phone stand*), SPP was superior in every aspect: *Deviation*, *task completion time* and *reported ease*, almost exclusively $p < 0.05$, the only exception being the *Cube*'s *deviation* ($p > 0.064$). Refer to Figures 6.5, C.1, C.2 for further detail. With the *Phone stand*, the differences in *deviation* were stronger than for the *Cube*. Two factors which caught our eye regarding this difference were the following:

Choice of the second profile shape — When using S2P with the *Cube*, it was the common approach to copy the profile (29/36 times). With *Phone stand*, the participants of-

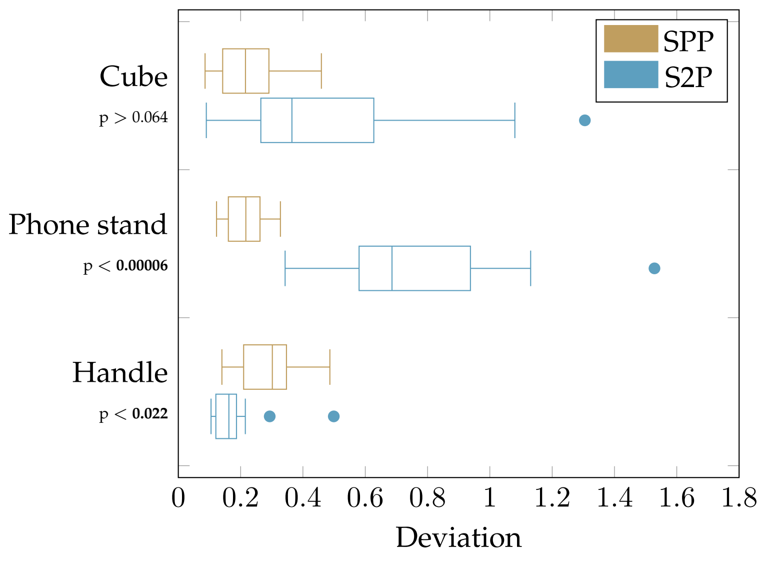


Figure 6.5: Distribution of *deviation* for each combination of model and technique. P-values from Wilcoxon ranked-sum test.

ten approximated the second profile by a triangle (26/36 times) (ref. Table 6.1). In this case, even if the vertices were perfectly aligned, the resulting model would have become skewed due to our calculation (ref. 5.3 "Two Profiles"). 3 participants uttered concern about our method of calculating centers, one participant explicitly suggesting, that the triangle approach *should* deliver the correct result if the corners are aligned with the bottom profile.

Bad results with S2P may have been partially caused by our misleading calculation.

Frequent S2P mistakes on first try — We often noticed very poor results in cases where it was the participant's overall first try with S2P (Figure 6.6). Likely resulting from incorrect depth estimation, the second profiles were sometimes so shifted in Z, that our algorithm created a curved connection instead of a straight one. 6 participants commented on the difficulty of aligning the profiles correctly due to a lack of depth perception.

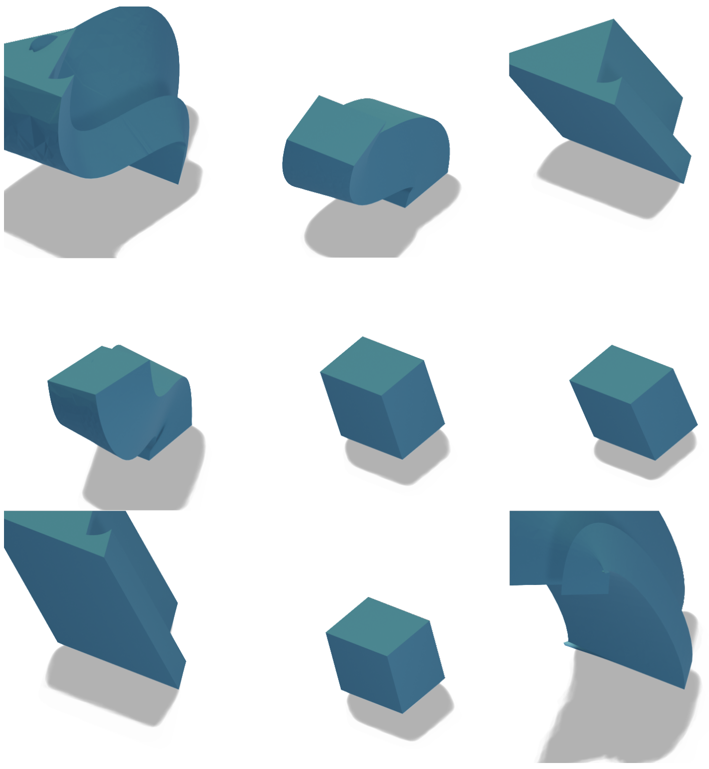


Figure 6.6: User-created models from instances in which either *Handle* or *Phone stand* was the first trial using S2P.

Curved Sweeps

S2P worked better with our curved sweep.

For the *Handle* model, S2P performed better with statistical significance in terms of *deviation* ($p < 0.022$), *task time* ($p < 0.0006$) and *reported ease* ($p < 0.003$). Note, that our *deviation* metric did not even take into account the shape of the *Handle*'s bow. Figure 6.7 shows all third attempts of the participants for creating the *Handle* via S2P. It became evident, that our participants had trouble creating the right curve for the extrusion path. What often confused our participants about our method of defining a spline by fitting it through control points, was the fact that each new point was able to retroactively alter the shape. This happened, because our fitting algorithm aimed for curvature continuity at the control points. Users therefore had to think ahead when placing the control points, as a segment of a curve

Users had trouble to create B-splines using our method.

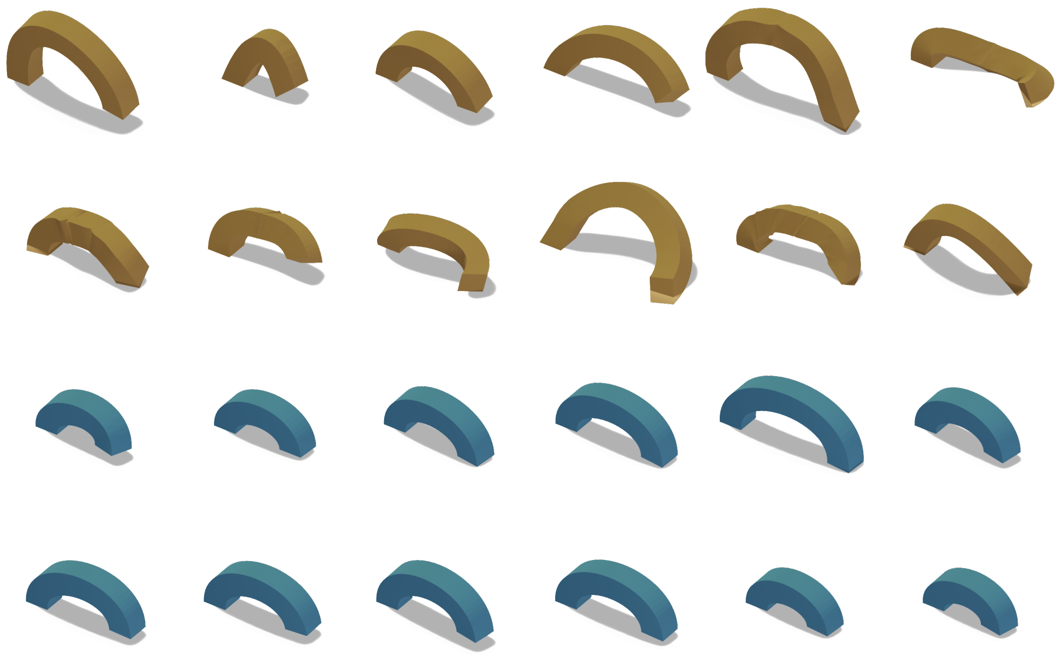


Figure 6.7: All third *Handle* attempts using SPP (orange) and S2P (blue).

was never truly fixed. Some circumvented this by placing a lot of control points (only 2 participants used four—the "sample solution") in small intervals, some doing so by holding the button.

The lack of control over the bow shape with S2P bothered some participants.

While a good bow shape was guaranteed with S2P, our participants were glad that it turned out the way it was meant to, but 4 participants uttered concern about their uncertainty if it would. 2 participants asked, if it was possible to add intermediate profiles in order to further guide the trajectory.
Aside from the bow shape, we noticed two main factors which may have lead to S2P's superiority:

Second profile choice — Having a simple profile shape, the most popular method of defining the second profile was to copy the initial profile (30/36 times, ref. Table 6.1), which already worked in favor of our calculation with the *Cube* model.

Target orientation — S2P enabled the participants to easily specify the orientation of the target profile. It therefore aligned well with the table in most cases, whereas with SPP our participants struggled to make it level (ref. Figure 6.7). With most participants, we observed a strong confusion about the skewness of the *Handle*'s ends using SPP. Note however, that the skewness was not considered in our *deviation* measure, which is why the lower *deviation* with S2P can not be attributed to physical guidance. The table provided a physical guide for SPP as well.

All participants preferred SPP in general.

It was almost a consensus between participants, that they preferred SPP for *Cube* and *Phone stand*, while preferring SPP for the *Handle*. When asked for a general preference, 100% of participants chose SPP, which was often justified by simply having to draw less.

Model	Method	Uses
Cube	Copy	29
	Triangle	7
Phone stand	Triangle	26
	Approximation	6
	Rectangle	3
	Copy	1
Handle	Copy	30
	Triangle	5
	Circle	1

Table 6.1: Usage frequencies of second profile methods per model for S2P.

Revolve

The revolution tasks were perceived as comparatively easy (ref. Figure C.5). Thus, the participants did not spend much time per task (ref. Figure C.4) and made fewer remarks than during the sweeping part. For the revolution techniques, our analysis will therefore be more focused on the quantitative data.

Model Quality

As described previously, the model quality was measured as *radius deviation* and *angle deviation*. In terms of *radius deviation*, RPC and R2P performed very similarly, while RPA slightly fell behind for both target models (Figure C.3, no statistical significance). As 4 participants noted, this might be due to the fact that in RPC and R2P the user is able to specify the diameter directly, whereas in RPA they had to define the radius.

Regarding *angle deviation*, RPA performed worse than both other techniques as well, with statistical significance except for in comparison with R2P for the *Door stopper* model ($p > 0.52$) (Figure 6.8). Likewise, RPC surpassed its competitors for *angle deviation* with statistics significance except for in comparison with R2P for the *Flower pot* model. The obvious explanation for RPC's good performance is, that the participants were able to draw the circle on top of the table, leaving tracking inaccuracies as the only cause for skewness.

Being able to draw the circle on the table stabilized the sketching.

We were surprised, that RPA performed significantly worse than R2P for the *Flower pot* model ($p < 0.001$), even though with RPA users only had to draw an orthogonal line, while with R2P they had to make it skewed by just the right amount. With the *Door stopper*, this effect did not occur. We propose, that because RPA's axis had to be drawn very close to the *Flower pot*'s curvy profile, it created an optical illusion.

Regarding the second profile shape in case of R2P—similarly to S2P—our participants went mostly for a minimal input. Approximating the second profile by a line was the clearly dominant approach (Figure 6.2).
Our data does not show significant differences in terms of *task time* (Figure C.4) and *reported ease* (Figure C.5) between the techniques.
The subjective ranking of the revolution techniques reveals the order RPA > RPC > R2P (Figure 6.9). This is interesting because—as shown earlier—the models created using RPA almost always exhibited significantly inferior quality. We will take up on this point in the discussion (Section 6.8).

In sum, users preferred RPA, despite its inferior performance.

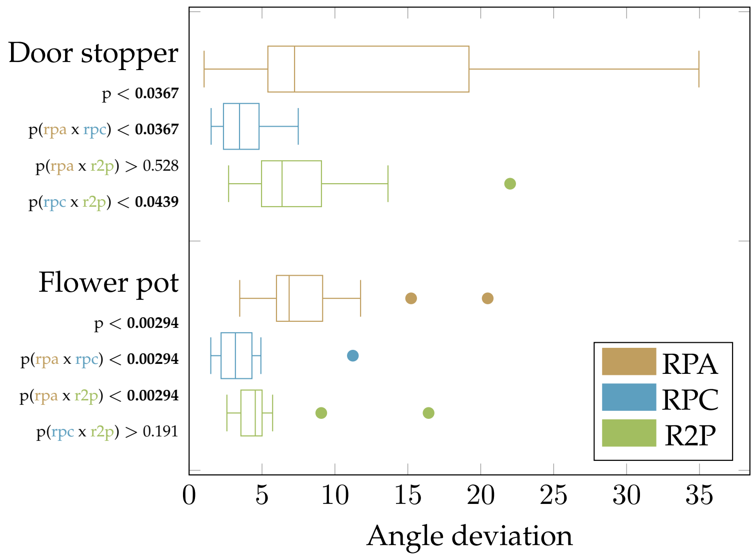


Figure 6.8: Distribution of *angle deviation* values for each combination of model and technique. Overall p-value calculated from Kruskal-Wallis test. Pairwise p-values from Wilcoxon ranked-sum test with Bonferroni correction.

General

Throughout the study, we collected participant's remarks and noticeable behavior. Inspired by the *Grounded Theory* approach ([Glaser and Strauss, 2017]), we assigned concepts to each observation and searched for patterns in order to generate clusters. An overview of the result can be seen in Table 6.3.

In the following, we will through our findings one cluster at a time, namely *depth perception/motor ability*, *visual guides*, *virtual constraints* and *hardware*. We will skip the cluster about *sweep specification*, as it has already been dealt with in depth in Section 6.7.
Note, that each statement could be mapped to multiple concepts and therefore appear in multiple categories.

Model	Method	Uses
Door stopper	Line	31
	Copy	2
	(Missing data)	3
Flower pot	Line	26
	Approximation	5
	Copy	2
	(Missing data)	3

Table 6.2: Usage frequencies of second profile methods per model for R2P.

Cluster	Concept	Participants
Depth perception	Problems with depth perception	11
	Preference for drawing on physical surfaces	7
Visual guides	Difficult size estimation	8
	Demand for live preview	7
	Demand for grids/measurements	5
Virtual constraints	Problems to close a stroke mid-air	10
	Appreciation of efficiency	9
	Demand for more snapping	4
	Demand to enforce orthogonal sweeping	4
	Demand for drawing on existing virtual objects	2
Sweep specification	Problems with spline creation	9
	Uncertainty about sweep generated by S2P	4
	Uncertainty about angle at SPP's end	3
	Uncertainty about S2P's center calculation	3
	Demand to add intermediate profiles to S2P	2
Hardware	Problems with keeping pen in viewport	9
	Back-and-forth to screen with pen hand was tedious	4
	Demand for AR glasses	3
	Discomfort due to phone heat	3
	Fatigue	2
	User tries to focus camera by tapping on screen	2

Table 6.3: The concepts we identified in the user feedback and observations. Concepts that were only observed once are not included.

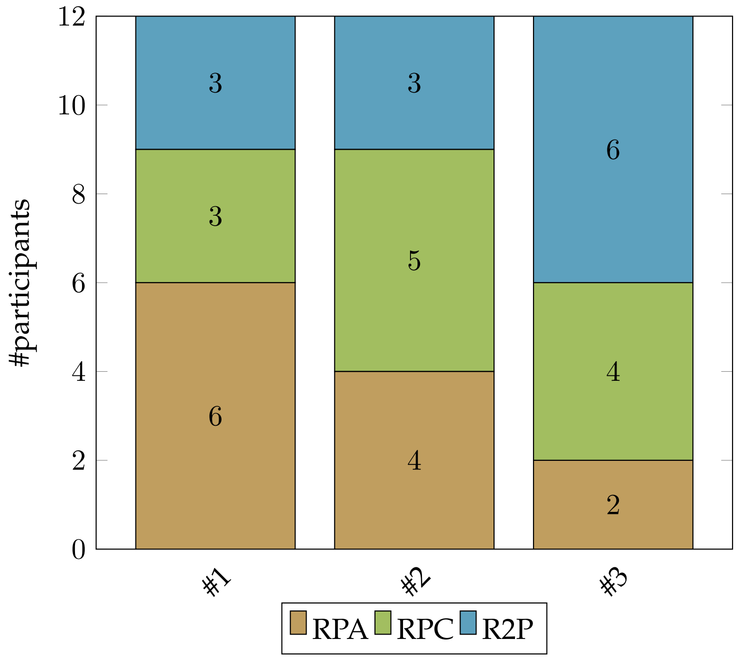


Figure 6.9: Subjective ranking of revolution techniques.

Depth Perception/Motor ability

Even with our minimalistic gestures, depth perception/motor ability remained a large issue.

Many observations were linked to the topic of deficient depth perception. 10/12 times a participant tried to create a closed path in mid-air during the free training sessions, she missed the starting point due to an offset in the camera's view direction and even displayed notable trouble in resolving the issue. The results of the sweeping study also revealed those issues, especially with SPP. A preference for drawing on physical surfaces was mentioned by 7 users. 3 participants wished for AR glasses, though only 1 justified it directly by the ability to have 3D vision.

Visual Guides

Live previews were frequently requested.

Partially related to the problem with depth perception, there was a strong demand for visual guides. The issue with estimating sizes was mentioned by 8 participants. The two popular solution approaches our participants mentioned were a live preview of the object to be created by the operations (7 users) and simply displaying a grid and/or measurements (5 users).

Virtual Constraints

Not only did our users want so see certain measurements, they also wanted them to be enforced. The demand for some type of virtual constraints was another frequently mentioned point. 10 user made remarks, wishing for various ways of snapping. In 4 cases this was related to snapping to existing points, either to close a path or to continue a path precisely from an existing point. Another 4 times it was related to "angular snapping", i.e. making sure that an angle is 90 degrees. Also related to snapping was the wish to being able to draw directly on existing (virtual) objects (2 users).

Hardware

The hardware category is broad, but we decided to not subdivide it further, as evaluating the hardware was not the main focus of this work. A lot of participants struggled with the bi-manual interaction style. 9 users struggled/forgot to keep the marker cube inside the camera's frustum at least once during the study. One participant however said that she quickly learned to keep it in view. Only 2 users complained about fatigue after the study, however 4 users said that it was tedious to move back-and-forth between the device screen and the scene with the pen hand. 3 users therefore wanted to have the undo functionality on the pen instead.
Participants frequently noticed the increasing heat of the phone after several trials. 3 explicitly described it as uncomfortable.

Misc

Parcicipants seldomly moved the pen hand during task execution.

In several ways, we observed a certain “movement laziness” among the participants. First, there was the aforementioned dislike of pressing the on-screen undo button. More prominently however, we noticed an unwillingness to move the phone hand and/or the head and upper body: In under 8% of all trials (33/432) did a participant move the phone to view the scene from another angle during task execution. 5 participants did not change their viewpoint even once during task execution throughout the entire study. One participant who moved the phone comparatively often and eventually stopped doing so, said, that she felt she achieved better results in less time by just relying on his muscle sense. Our data shows no indication that the amount of viewpoint switches had a systematic effect on the quality of the outcome. However, these observations are to be treated with caution, as the tasks were not set out to particularly benefit from altering the viewpoint and our study was generally not designed with this variable in mind.

6.8 Discussion

The “traditional” approaches were popular among our users.

When it comes to user preference, our participants overall favored the traditional techniques for both general sweeps and revolutions. We see three plausible explanations for this:

1. In most cases, Sweep (Profile + Path) and Revolve (Profile + Axis) required the simplest input—often just a straight vertical line.
2. SPP and RPA were a direct mapping from the sketch input to the executed action, while Sweep (Two Profiles), Revolve (Profile + Circle) and Revolve (Profile + Axis) all featured some sort of “auto-correction/auto-completion”.
3. There was relatively large 3D modeling experience

among our participants, making SPP and RPA feel more familiar.

To further investigate argument 3, we tried to test the hypothesis: "People who vote RPA first place have used more 3D modeling tools in the past than those who do not". However, the sample size was small and a one-sided Mann-Whitney U test yielded $p > 0.26$. Notably though, the average number of CAD tools used by our participants who liked RPA best, was even lower than for those who did not (1.8 vs 2.7).
Despite not being preferred in general, S2P and RPC showed very promising results in situations where physical constraints could be exploited. However, the same physical constraints could have worked in favor of the other techniques, e.g. had the profiles been lying on the table in the revolution study. Conversely, RPC will likely be unsuited for pure mid-air modeling, given the issues our participants had with even creating a closed path mid-air. RPA might fail in situations, in which a real-world object should be copied by tracing it, because the rotation axis would lie inside of the object. Summarizing, it is obvious that it depends heavily on the situation, which technique is favorable.

Physical constraints can work both in favor, and against a certain techniques.

As we perceived a scepticism towards auto-completions and -corrections especially for sweeping via S2P, we think it might be a good idea to change S2P into a general combination of sweeping and lofting, meaning that the profile shape interpolates between the profiles, and the path can be specified manually. This would also add a range of new interesting applications for this method.
Given that simple extrusions in normal direction seem to be way more common in everyday modeling than general sweeps, modern CAD tools probably do the right thing in separating those actions in favor of an easier extrusion operation. This is also what some of our participants wished for.

Correcting the user input should be treated with caution.

There was a lot of demand for live previews and visual guides, hinting that while our users were sketching, they were insecure about how the objects would be generated. Live previews seem to be useful indeed, but the implemen-

A live preview may hide important virtual and real-world content.

tation might become challenging due to hardware limitations. We advise to not render live previews opaque, as the current sketch and especially real-world objects could be hidden by them. Finding the right types of visual guides and integrating them in a way that the UI does not get bloated will be a task for future work.

Making more intelligent use of physical surfaces seems more promising than drawing projectedly on virtual ones mid-air.

Though we tried to mitigate the problems related to bad depth perception by using solid modeling techniques, it seems to remain a big issue. Especially the occasionally requested—and definitely desirable—ability to draw onto virtual surfaces is hard to achieve using pure mid-air interaction. However, [Arora et al., 2017] already attested that the obvious solution, to draw projectedly onto a surface, will not result in precision comparable to drawing on physical surfaces. Instead, they propose mapping the virtual surfaces onto physical boards, which are either handheld (i.e. surface adjusts to physical one) or actuated via robotic arms (i.e. physical surface adjusts to virtual one). With the ARPen however, the non-dominant hand is occupied and robotic arms are not portable. We think it would be worthwhile to investigate the option of mapping physical surfaces onto existing ones in the vicinity, similarly to how it was possible in *Mockup Builder* ([Ara, 2012]), but not restricted to the table top. Mapping them to the phone's screen and drawing on it would also be an option more leaning towards what [Arora et al., 2017] suggested, but our findings suggest that frequent switching between working in-situ and on-screen could be tedious.

B-splines passing through control points did not work well.

Regarding the sketching itself, it became evident that our B-spline method did not appeal to design novices.
Lastly, there need to be ways to rotate and scale the scene. Even disregarding our observation that people seem to be unmotivated to physically move around the scene, looking at it from certain angles may be sheer impossible due to objects blocking the way.

Limitations

Some factors may have had a negative impact on the validity of our results:

We discarded the data of participant 6 due to incompleteness and included user 13 in the study instead, without adjusting our Latin square. It therefore lacks the 6th sequence, but contains the 1st sequence twice. The impact of this on the results is unclear, but as 10 out of 12 sequences were correct, we hope that it was neglectable.

Two left-handed participants were included in the study. When asked if something felt inconvenient, one of them mentioned the placement of the pen's hardware buttons and she was at one point irritated by the USB cable.

Except for one case, all participants had at least used one 3D modeling tool in the past (mean: 2.25, sd: 1.71). Therefore, our users were probably more experienced than the average person.

Chapter 7

Summary and Future Work

7.1 Summary and Contributions

In this thesis we developed a framework for the ARPen that enables the creation and manipulation of volumetric geometry via solid modeling and constructive solid geometry techniques. We aimed for a minimalist and simple API. Still, with Open CASCADE Technology in the background, it is possible to extend the functionality way beyond, leveraging the full potential of a professional CAD modeling kernel.

For the supported modeling operations, we implemented a set of different techniques by which to use them. Summing up, there were two techniques for creating general sweeps, three for revolving, two for Boolean operations and one for lofting. We used the techniques as a basis to explore sketch-based geometry creation with the ARPen in a user study. However, due to time constraints, the user study focused only on general sweeps and revolutions.

Our observations once more stress the crucial role of physical constraints in mir-air—especially in-situ—modeling. Based on the situation, they can work in favor of, but also

against a certain technique. We therefore strongly advice to provide users with a diverse set of input techniques.
Developers should also keep in mind the user's desire to work efficiently. This applies both to the efficiency of the modeling techniques, as well as to the reduction of expansive hands and body movement in general.
Our input method based on generating geometry only after the sketching was finished, limited the user's awareness of their action's outcome. Live previews and other visual guides should be considered.

Summing up, we generated empirical insights into the space of in-situ 3D modeling in smartphone AR. Our findings may be used to guide the development of such systems in the future. Additionally, our implementation has plenty of potential to be used in future studies.

7.2 Future Work

Test remaining functionality.

Due to time constraints, we were not able to include all functionality we had implemented in the user study. The first obvious option for future work would therefore be to evaluate the remaining functions, namely lofting and Boolean operations.

Alternative curve creation.

Judging by our observations, work has to be done on the sketching itself. While our approach with multi-point lines diminished the issue of high jittering, our method of curve creation was not well received.

Mapping virtual surfaces to physical ones.

Track pen differently.

As also mentioned before, it would be worthwhile to investigate on ways in which to project virtual surfaces onto real-world ones, in order to make efficient use of physical constraints. Alternative ways of tracking the pen may be considered (e.g. [Yoon et al., 2016]), regarding the apparent difficulties users have with making sure that the marker cube stays within the camera's frustum.

Add 3D scanning.

A promising addition to the ARPen concept, would be the ability to 3D scan real-world objects and thereby incorporate them into the design, e.g. in order to create fitting cavities using Boolean operations as demonstrated in [Weichel et al., 2014].

Consider retrieval-based approaches.

In our work, we excluded retrieval-based techniques as they stood out too much from the other techniques, which were focused on creating custom designs from scratch. However, regarding the large existing repositories of designs, sketch-based retrieval may actually be interesting to look at. The ARPen could be used to not only ease the browsing, but also to scale and adjust the designs as desired without the need for taking measurements.

Appendix A

Participant Information Questionnaire

Participant Information

* Erforderlich

1. **User Id** *

2. **Your age**

3. **Your Gender**
   Markieren Sie nur ein Oval.
   - ◯ Male
   - ◯ Female
   - ◯ Prefer not to say
   - ◯ Sonstiges: ______

4. **Your dominant hand is**
   Markieren Sie nur ein Oval.
   - ◯ Right
   - ◯ Left
   - ◯ Both

5. **Which tools for 3D modeling did you already use? (Leave empty if nothing applies)**
   Wählen Sie alle zutreffenden Antworten aus.
   - ☐ Blender
   - ☐ Cinema 4D
   - ☐ Autodesk 3ds Max
   - ☐ Autodesk Maja
   - ☐ AutoCAD
   - ☐ Autodesk Inventor
   - ☐ Autodesk Fusion 360
   - ☐ TinkerCAD
   - ☐ FreeCAD
   - ☐ OpenSCAD
   - ☐ SolidWorks
   - ☐ CATIA
   - ☐ Autodesk Mudbox
   - ☐ ZBrush
   - ☐ Sonstiges: ______

6. **Did you already work with the ARPen?**
Markieren Sie nur ein Oval.

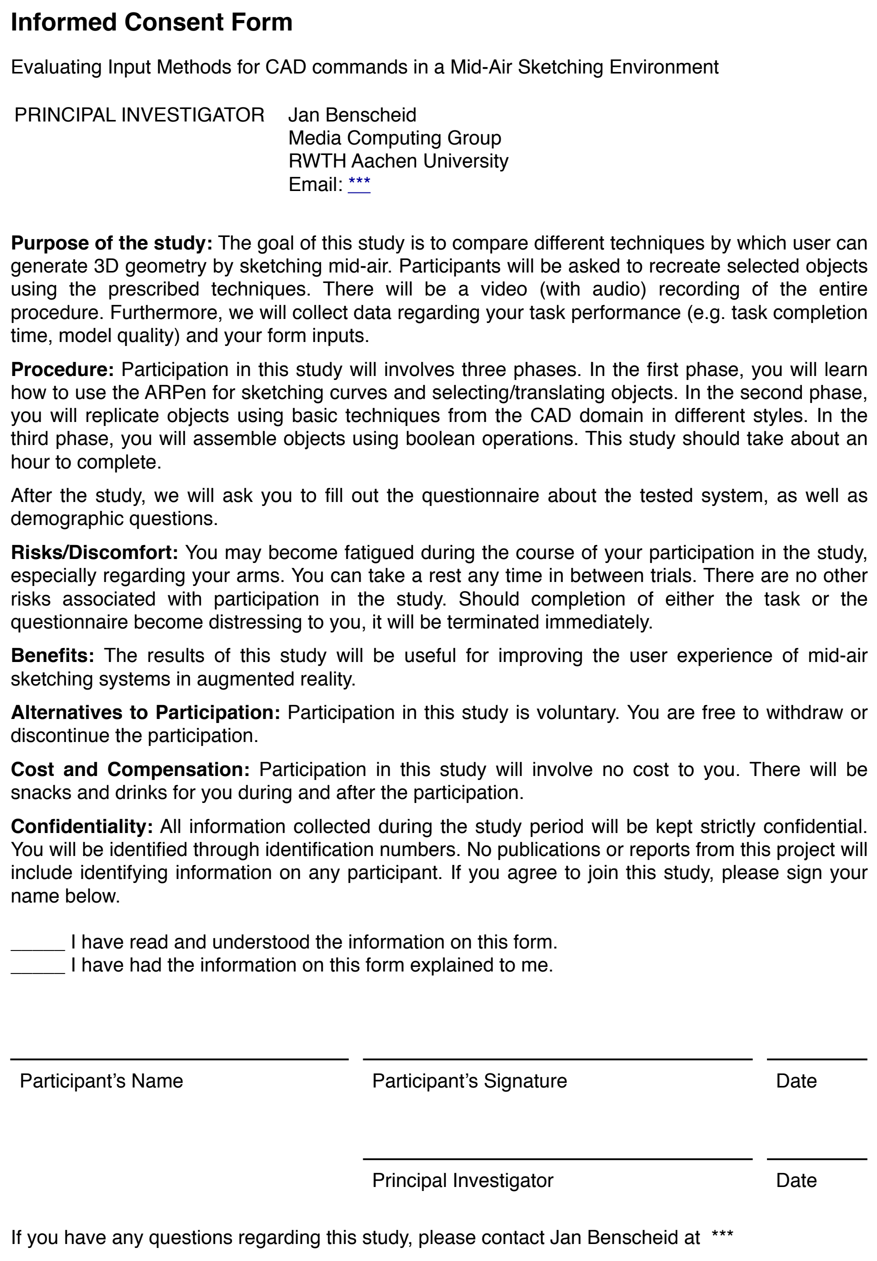
- ⬭ Yes
- ⬭ No

7. **Do you have experience with other augmented reality systems or applications?**

____

Appendix B

User Study Consent Form



Informed Consent Form

Evaluating Input Methods for CAD commands in a Mid-Air Sketching Environment

PRINCIPAL INVESTIGATOR	Jan Benscheid Media Computing Group RWTH Aachen University Email: ***

Purpose of the study: The goal of this study is to compare different techniques by which user can generate 3D geometry by sketching mid-air. Participants will be asked to recreate selected objects using the prescribed techniques. There will be a video (with audio) recording of the entire procedure. Furthermore, we will collect data regarding your task performance (e.g. task completion time, model quality) and your form inputs.

Procedure: Participation in this study will involves three phases. In the first phase, you will learn how to use the ARPen for sketching curves and selecting/translating objects. In the second phase, you will replicate objects using basic techniques from the CAD domain in different styles. In the third phase, you will assemble objects using boolean operations. This study should take about an hour to complete.

After the study, we will ask you to fill out the questionnaire about the tested system, as well as demographic questions.

Risks/Discomfort: You may become fatigued during the course of your participation in the study, especially regarding your arms. You can take a rest any time in between trials. There are no other risks associated with participation in the study. Should completion of either the task or the questionnaire become distressing to you, it will be terminated immediately.

Benefits: The results of this study will be useful for improving the user experience of mid-air sketching systems in augmented reality.

Alternatives to Participation: Participation in this study is voluntary. You are free to withdraw or discontinue the participation.

Cost and Compensation: Participation in this study will involve no cost to you. There will be snacks and drinks for you during and after the participation.

Confidentiality: All information collected during the study period will be kept strictly confidential. You will be identified through identification numbers. No publications or reports from this project will include identifying information on any participant. If you agree to join this study, please sign your name below.

_____ I have read and understood the information on this form.
_____ I have had the information on this form explained to me.

Participant's Name	Participant's Signature	Date
	Principal Investigator	Date

If you have any questions regarding this study, please contact Jan Benscheid at ***

Figure B.1: Consent form of the user study. E-mail address removed for publication.

Appendix C

Diagrams

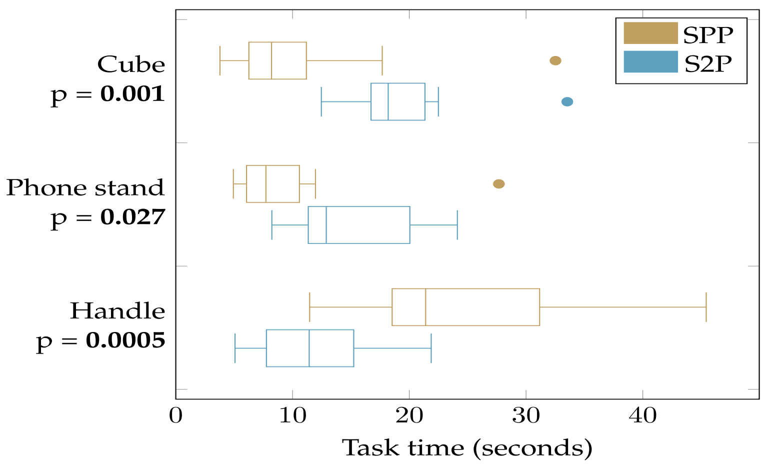


Figure C.1: Distribution of *task times* for each combination of model and technique with sweeping. P-values from Wilcoxon ranked-sum test.

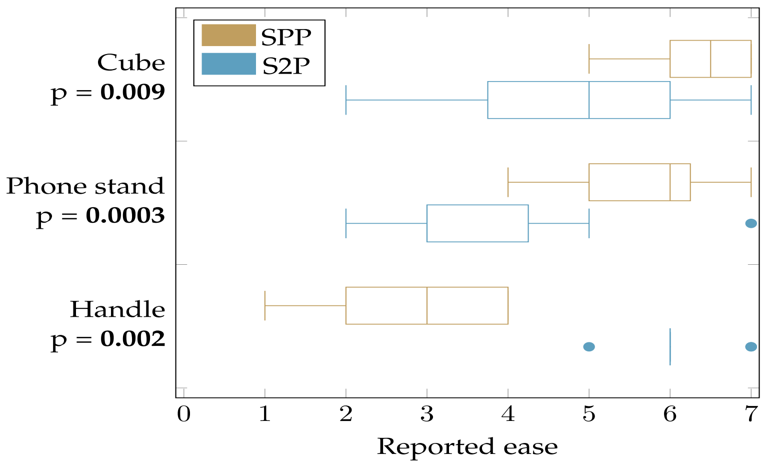


Figure C.2: Distribution of *reported ease* values for each combination of model and technique with sweeping. P-values from Wilcoxon ranked-sum test.

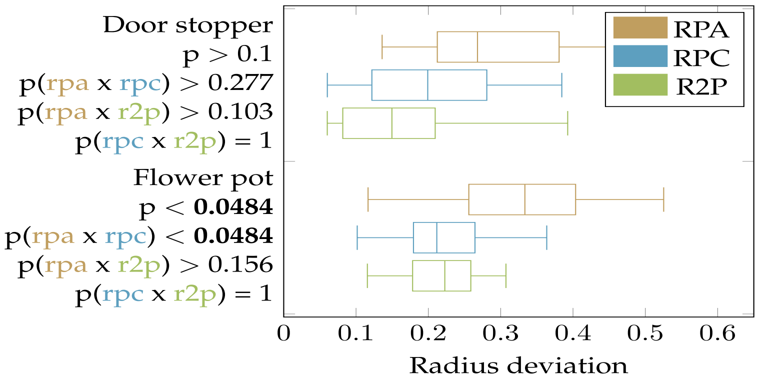


Figure C.3: Distribution of *radius deviation* values for each combination of model and technique with revolving. Overall p-value calculated from Kruskal-Wallis test. Pairwise p-values from Wilcoxon ranked-sum test with Bonferroni correction.

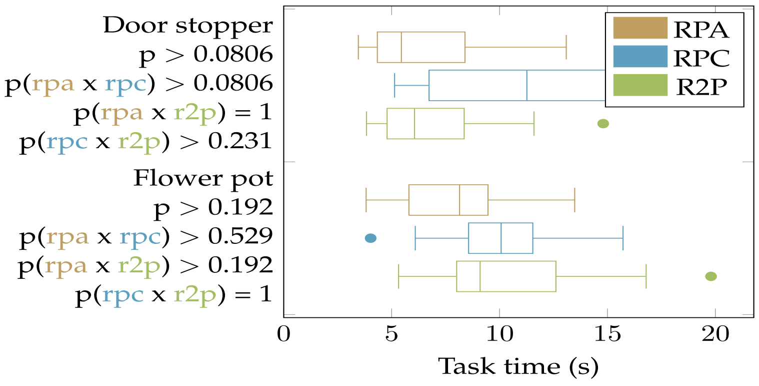


Figure C.4: Distribution of task times for each combination of model and technique with revolving. Overall p-value calculated from Kruskal-Wallis test. Pairwise p-values from Wilcoxon ranked-sum test with Bonferroni correction.

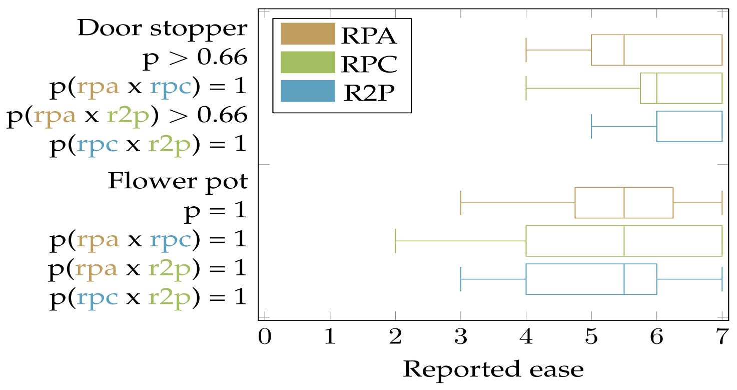


Figure C.5: Distribution of reported ease values for each combination of model and technique with revolving. Overall p-value calculated from Kruskal-Wallis test. Pairwise p-values from Wilcoxon ranked-sum test with Bonferroni correction.

Appendix D

User Study Questions

D.1 Post-Techniques Questions

The following questions were asked once after finishing the sweeping- and the revolution part of the study. The participant was always asked to justify their response. It was also allowed to give different answers depending on the target model.

- With which technique did you find it easier to come up with the necessary inputs?
- Which technique was easier to apply?
- Rank the techniques according to your general preference.
- Do you have suggestions for an alternative approach?

D.2 Post-Study Questions

These questions were asked at the very end of the study:

- How did you get along with the system in general?

- Are there features you would wish for?
- Do you have any additional remarks?

Bibliography

Michael Alba. What's the Difference Between Parametric and Direct Modeling?, June 2018. URL `https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/16587/Whats-the-Difference-Between-Parametric-and-Direct-Modeling.aspx`.

David Anderson, James L. Frankel, Joe Marks, Aseem Agarwala, Paul Beardsley, Jessica Hodgins, Darren Leigh, Kathy Ryall, Eddie Sullivan, and Jonathan S. Yedidia. Tangible Interaction + Graphical Interpretation: A New Approach to 3d Modeling. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 393–402, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 978-1-58113-208-3. doi: 10.1145/344779.344960. URL `http://dx.doi.org/10.1145/344779.344960`.

Bruno R De Ara. Mockup builder: direct 3d modeling on and above the surface in a continuous interaction space. page 8, 2012.

Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George Fitzmaurice. Experimental Evaluation of Sketching on Surfaces in VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, pages 5643–5654, Denver, Colorado, USA, 2017. ACM Press. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025474. URL `http://dl.acm.org/citation.cfm?doid=3025453.3025474`.

Rahul Arora, Rubaiat Habib Kazi, Tovi Grossman, George Fitzmaurice, and Karan Singh. SymbiosisSketch: Combining 2d & 3d Sketching for Designing Detailed 3d Objects in Situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*, pages 1–15, Montreal QC, Canada, 2018. ACM Press. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173759. URL `http://dl.acm.org/citation.cfm?doid=3173574.3173759`.

Oliver Bimber, L. Miguel Encarnação, and André Stork. A multi-layered architecture for sketch-based interaction within virtual environments. *Computers & Graphics*, 24: 851–867, 2000. doi: 10.1016/S0097-8493(00)00088-1.

Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc. T. Olano. 3dm: a three dimensional modeler using a head-mounted display. In *Proceedings of the 1992 symposium on Interactive 3D graphics - SI3D '92*, pages 135–138, Cambridge, Massachusetts, United States, 1992. ACM Press. ISBN 978-0-89791-467-3. doi: 10.1145/147156.147182. URL `http://portal.acm.org/citation.cfm?doid=147156.147182`.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3d Model Retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003. ISSN 1467-8659. doi: 10.1111/1467-8659.00669. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00669`.

I. D. Coope. Circle fitting by linear and nonlinear least squares. *Journal of Optimization Theory and Applications*, 76(2):381–388, February 1993. ISSN 1573-2878. doi: 10.1007/BF00939613. URL `https://doi.org/10.1007/BF00939613`.

Martin L Culpepper. 2.972 Solid Modeling. URL `https://web.mit.edu/2.972/www/solid_modeling.html`.

Mike De la Flor and Bridgette Mongeon. *Digital sculpting with Mudbox: essential tools and techniques for artists*. Focal Press, Burlington, MA, 2010. ISBN 978-0-240-81203-8. OCLC: ocn606053101.

Lynn Eggli, Ching-yao Hsu, Beat Bruderlin, and Gershon Elber. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design*, 29:101–112, February 1997. doi: 10.1016/S0010-4485(96)00039-5.

Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics*, 31(4):1–10, July 2012. ISSN 07300301. doi: 10.1145/2185520.2185527. URL `http://dl.acm.org/citation.cfm?doid=2185520.2185527`.

Barney G. Glaser and Anselm L. Strauss. *Discovery of Grounded Theory : Strategies for Qualitative Research*. Routledge, July 2017. ISBN 978-0-203-79320-6. doi: 10.4324/9780203793206. URL `https://www.taylorfrancis.com/books/9780203793206`.

Luisa Hoffmann. *Eliciting User Gestures for the Interaction with a Pen-Based 3D Modeling Device*. Bachelor's Thesis, RWTH Aachen University, Aachen, March 2017.

Ke Huo, Vinayak, and Karthik Ramani. Window-Shaping: 3d Design Ideation by Creating on, Borrowing from, and Looking at the Physical World. In *Proceedings of the Tenth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '17*, pages 37–45, Yokohama, Japan, 2017. ACM Press. ISBN 978-1-4503-4676-4. doi: 10.1145/3024969.3024995. URL `http://dl.acm.org/citation.cfm?doid=3024969.3024995`.

Takeo Igarashi and John F. Hughes. A Suggestive Interface for 3d Drawing. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 173–181, New York, NY, USA, 2001. ACM. ISBN 978-1-58113-438-4. doi: 10.1145/502348.502379. URL `http://doi.acm.org/10.1145/502348.502379`. event-place: Orlando, Florida.

Bret Jackson and Daniel F. Keefe. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3d Models in VR. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1442–1451, April 2016. ISSN 1077-2626. doi: 10.1109/TVCG.2016.2518099. URL `http://ieeexplore.ieee.org/document/7383322/`.

Yuna Kang, Hyungki Kim, Hiromasa Suzuki, and Soonhung Han. Feature-based 3d CAD modeling on smart device using multi-touch gesture. In *2012 Asian Conference on Design and Digital Engineering*. Society of CAD/CAM engineers, 2012.

Daniel F. Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H. Laidlaw, and Joseph J. LaViola. CavePainting: a fully immersive 3d artistic medium and interactive experience. In *Proceedings of the 2001 symposium on Interactive 3D graphics - SI3D '01*, pages 85–93, Not Known, 2001. ACM Press. ISBN 978-1-58113-292-2. doi: 10.1145/364338.364370. URL `http://portal.acm.org/citation.cfm?doid=364338.364370`.

Dae Hyun Kim and Myoung-Jun Kim. A new modeling interface for the pen-input displays. *Computer-Aided Design*, 38(3):210–223, March 2006. ISSN 00104485. doi: 10.1016/j.cad.2005.10.007. URL `https://linkinghub.elsevier.com/retrieve/pii/S0010448505001818`.

Ernst Kruijff, J. Edward Swan, and Steven Feiner. Perceptual issues in augmented reality revisited. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 3–12, Seoul, Korea (South), October 2010. IEEE. ISBN 978-1-4244-9343-2. doi: 10.1109/ISMAR.2010.5643530. URL `http://ieeexplore.ieee.org/document/5643530/`.

Jinggao Li, Soonhung Han, Suchul Shin, Seunghoon Lee, Yuna Kang, Hayoung Cho, Hyungki Kim, Ilhwan Song, Ikjune Kim, and Pranveer Singh Rathore. CAD Data Exchange Using the Macro-Parametrics Approach: An Error Report. 10, December 2010.

Paul Milgram and Fumio Kishino. A Taxonomy of Mixed Reality Visual Displays. 1994.

Bojan Milosevic, Flavio Bertini, Elisabetta Farella, and Serena Morigi. A SmartPen for 3d interaction and sketch-based surface modeling. *The International Journal of Advanced Manufacturing Technology*, September 2015. ISSN 0268-3768, 1433-3015. doi: 10.1007/s00170-015-7828-1. URL `http://link.springer.com/10.1007/s00170-015-7828-1`.

João Pereira, Joaquim Jorge, Vasco Branco, Nelson Silva, Tiago Cardoso, and Fernando Nunes Ferreira. Cascading Recognizers for Ambiguous Calligraphic Interaction. September 2004.

Kevin Ponto, Ross Tredinnick, Aaron Bartholomew, Carrie Roy, Dan Szafir, Daniel Greenheck, and Joe Kohlmann. SculptUp: A rapid, immersive 3d modeling environment. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 199–200, Orlando, FL, March 2013. IEEE. ISBN 978-1-4673-6098-2 978-1-4673-6097-5. doi: 10.1109/3DUI.2013.6550247. URL `http://ieeexplore.ieee.org/document/6550247/`.

Jeff Sauro and Joseph S. Dumas. Comparison of Three One-question, Post-task Usability Questionnaires. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1599–1608, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518946. URL `http://doi.acm.org/10.1145/1518701.1518946`. event-place: Boston, MA, USA.

Steven Schkolne, Michael Pruett, and Peter Schröder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '01*, pages 261–268, Seattle, Washington, United States, 2001. ACM Press. ISBN 978-1-58113-327-1. doi: 10.1145/365024.365114. URL `http://portal.acm.org/citation.cfm?doid=365024.365114`.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-View Convolutional Neural Networks for 3d Shape Recognition. pages 945–953, 2015. URL `https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Su_Multi-View_Convolutional_Neural_ICCV_2015_paper.html`.

William Vaughan. *Digital Modeling*. New Riders, Berkeley, CA, 2012. ISBN 978-0-321-70089-6. OCLC: ocn779131781.

Philipp Wacker, Simon Voelker, Adrian Wagner, and Jan Borchers. Physical Guides: An Analysis of 3d Sketching Performance on Physical Objects in Augmented Reality. In *Proceedings of the Symposium on Spatial User Interaction - SUI '18*, pages 25–35, Berlin, Germany, 2018. ACM Press. ISBN 978-1-4503-5708-1. doi: 10.1145/3267782.3267788. URL `http://dl.acm.org/citation.cfm?doid=3267782.3267788`.

Philipp Wacker, Oliver Nowak, Simon Voelker, and Jan Borchers. ARPen: Mid-Air Object Manipulation Techniques for a Bimanual AR System with Pen & Smartphone. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*, pages 1–12, Glasgow, Scotland Uk, 2019. ACM Press. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300849. URL `http://dl.acm.org/citation.cfm?doid=3290605.3300849`.

Fang Wang, Le Kang, and Yi Li. Sketch-Based 3d Shape Retrieval Using Convolutional Neural Networks. pages 1875–1883, 2015. URL `https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Wang_Sketch-Based_3D_Shape_2015_CVPR_paper.html`.

Felix Wehnert. *Pen-based Drawing in Augmented Reality on Mobile Phones*. Bachelor's Thesis, RWTH Aachen University, Aachen, April 2018.

Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, and Hans W. Gellersen. MixFab: a mixed-reality environment for personal fabrication. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pages 3855–3864, Toronto, Ontario, Canada, 2014. ACM Press. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557090. URL `http://dl.acm.org/citation.cfm?doid=2556288.2557090`.

Christian Weichel, John Hardy, Jason Alexander, and Hans Gellersen. ReForm: Integrating Physical and Digital Design through Bidirectional Fabrication. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*, pages 93–102, Daegu, Kyungpook, Republic of Korea, 2015. ACM Press. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.

2807451. URL `http://dl.acm.org/citation.cfm?doid=2807442.2807451`.

Gerold Wesche and Hans-Peter Seidel. FreeDrawer: A Free-form Sketching System on the Responsive Workbench. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, pages 167–174, New York, NY, USA, 2001. ACM. ISBN 978-1-58113-427-8. doi: 10.1145/505008.505041. URL `http://doi.acm.org/10.1145/505008.505041`. event-place: Baniff, Alberta, Canada.

E. Wiese, J. H. Israel, A. Meyer, and S. Bongartz. Investigating the Learnability of Immersive Free-hand Sketching. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 135–142, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association. ISBN 978-3-905674-25-5. URL `http://dl.acm.org/citation.cfm?id=1923363.1923387`. event-place: Annecy, France.

Min Xin, Ehud Sharlin, and Mario Costa Sousa. Napkin sketch: handheld mixed reality 3d sketching. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology - VRST '08*, page 223, Bordeaux, France, 2008. ACM Press. ISBN 978-1-59593-951-7. doi: 10.1145/1450579.1450627. URL `http://portal.acm.org/citation.cfm?doid=1450579.1450627`.

Sang Ho Yoon, Ke Huo, and Karthik Ramani. TMotion: Embedded 3d Mobile Input using Magnetic Sensing Technique. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction - TEI '16*, pages 21–29, Eindhoven, Netherlands, 2016. ACM Press. ISBN 978-1-4503-3582-9. doi: 10.1145/2839462.2839463. URL `http://dl.acm.org/citation.cfm?doid=2839462.2839463`.

Heinz Zllighoven. *Object-Oriented Construction Handbook: Developing Application-Oriented Software with the Tools & Materials Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2004. ISBN 978-1-55860-687-6.

Klen Čopič Pucihar, Paul Coulton, and Jason Alexander. Evaluating Dual-view Perceptual Issues in Handheld

Augmented Reality: Device vs. User Perspective Rendering. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pages 381–388, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2129-7. doi: 10.1145/2522848.2522885. URL `http://doi.acm.org/10.1145/2522848.2522885`. event-place: Sydney, Australia.

Index

Typeset October 14, 2019