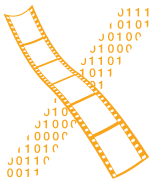

Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Parameterized Tangible Detection on Capacitive Screens

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

by
Aaron Krämer

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrich Schroeder

Registration date: 18.10.2017
Submission date: 29.11.2017

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matrikelnummer

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

_____

_____

_____

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

Approximately a decade ago, around 2007, capacitive touch screens were introduced to the consumer market. Their direct interaction with UI elements made it easier for many people to use such devices. But the planar surface is missing haptic experience and tactile feedback. To overcome this problem researchers developed tangibles. These are physical objects with capacitive markers on the bottom to enable recognition by touch screens. Many projects are using additional hardware to detect tangibles, like tracking technology attached to the screen or electronic components included in the tangible itself. A few approaches are working with passive tangibles just using the capacitive screen without any additional tracking technology. The problem is the identification since the only data available are the touches. Additional information like the rotation of the tangible as well as recovery algorithms are of interest, if a touch event is lost. Parameters based on the touch data will be investigated, analyzed and evaluated through a study. The results are integrated into a Swift SDK called PASTA which is capable of identifying passive tangibles as well as detecting their rotation.

Überblick

Vor ungefähr einem Jahrzent, um das Jahr 2007 haben kapazitive Touchscreens den Verbrauchermarkt erreicht. Die direkte Interaktion mit UI Elementen macht es für viele Menschen leichter mit solchen Geräten umzugehen. Die glatte Oberfläche bietet jedoch keine haptische Erfahrung oder taktiles Feedback. Um dieses Problem zu lösen arbeiten Wissenschaftler an sogenannte Tangibles. Dies sind physikalische Objekte mit kapazitiven Markern auf der Unterseite mit dessen Hilfe sie von einem Touchscreen erkannt werden. Viele Projekte benutzen zusätzliche Hardware um Tangibles zu erkennen, wie z.B. zusätzliche Tracking Technologien oder elektronische Komponenten im Tangible. Einige Ansätze arbeiten mit passiven Tangibles, das heißt nur mit dem verbauten kapazitiven Bildschirm und keiner zusätzlichen Tracking Technologie. Das Problem dabei ist die Identifizierung, da man nur die vom Tangible generierten Touch Events zur Verfügung hat. Zusätzlich sind Informationen wie die aktuelle Rotation des Tangibles von Interesse, sowie fehlerverzeihende Algorithmen falls ein Touch Event verschwindet. Basierend auf den Touch Events werden verschiedene Parameter mittels einer Studie untersucht, analysiert und evaluiert. Die Ergebnisse werden in ein Swift SDK integriert, welches in der Lage ist passive Tangibles zu identifizieren sowie die Rotation zu bestimmen.

Acknowledgements

I would like to thank Simon for the freedom throughout my thesis, Sebastian for some useful discussions about iOS, Krishna for providing me feedback about my statistical analyzes and the people of the HiWi-room (Kerstin, Ilja, Marcel, Oliver, ...) for all the funny moments and distracting discussions.

Thanks to my roommate Tanja for listening about my thesis moaning and for being a constant chat buddy. There are so many more people (Marie, Jan, Valerie, Matthias, Kathi, Dennis, Marcel, Sarah, ...) which were not directly involved in my thesis but influenced me in many ways. Thank you for being my friends!

Thanks to my brothers but especially to my parents who always believed in me, always supported me, and always helped me even if I was a little bit rude ;-) I love you!

Special thanks to Kerstin, Ilja, Tanja, Ben, and Marie for prove reading my thesis.

The comic below is for all of you.

Figure 1: Thanks to Joscha Sauer from NICHTLUSTIG allowing me to use this comic :-) Copyright of NICHTLUSTIG.

Conventions

Throughout this thesis we use the following conventions.

If a special term is used for the first time it will be emphasized e.g. the word *Kappes*.

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

KAPPES:
A word of the german dialect Kölsch. It can be translated as 'cabbage' or 'nonsense' with regard to the context.

Definition:
Kappes

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

```
myClass
```

Referenced Swift code is styled like in the Apple Developer Documentation [Apple, 2017a]. Type annotations and argument names are omitted. For example the function `hitTest(_ point: CGPoint, with event: UIEvent?) -> UIView?` is referenced as `hitTest(_:with:)`.

The whole thesis is written in American English.

Text is written in first person plural because of esthetic reasons.

Chapter 1

Introduction

Since the introduction of the Apple iPhone in 2007 the number of shipped smartphones increased heavily between 2009 and 2016 (Figure 1.1) with a shallower continuous growing trend. In the US, the number of smartphones used by mobile phone users reached 67% in 2014 and is predicted to be 92% in 2020 [Statista.com, 2017]. Not only the

Shipment and use of smartphones increased.

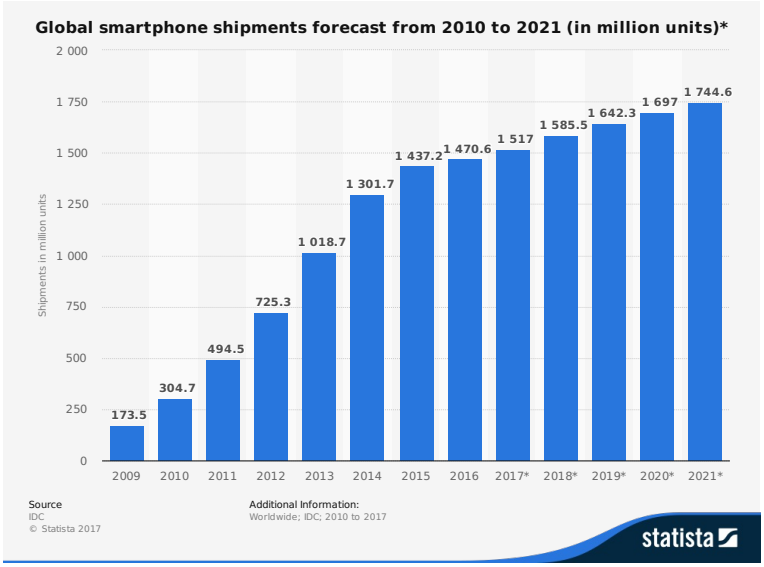

Figure 1.1: Global smartphone shipments 2010 - 2021. Years postfixed with * are forecasts.

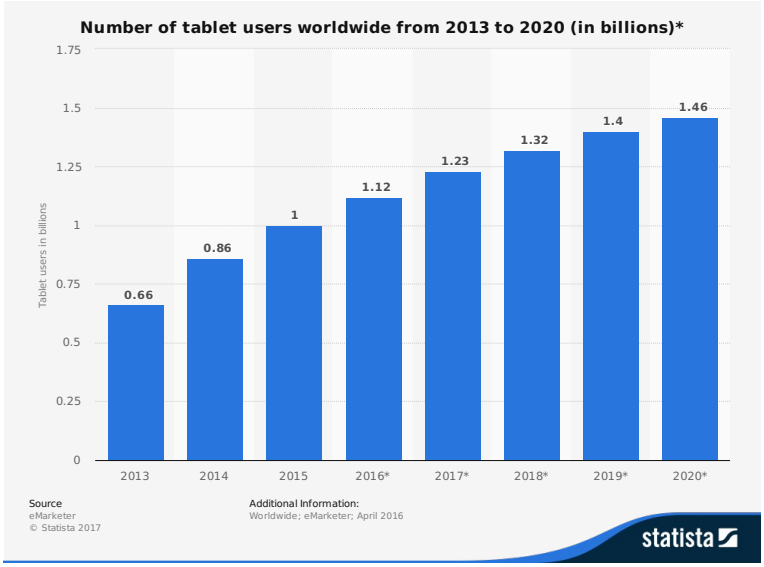

Figure 1.2: Number of tablet users worldwide 2013 - 2020. Years postfixed with * are forecasts.

Touch screens are used everywhere today.

use of touch screens in the mobile phone area increased but also the number of tablet users doubled within the last four years (Figure 1.2). The graph is also showing an upwards trend for the next years. Tablets may replace reading and writing on paper in the future [Statista.com, 2017] especially with the introduction of the Apple Pencil [Apple, 2017b]. Today, touch screens are used in almost every area, e.g., tablets for work in the field [YouTube, 2017a]. Tesla replaced many of the physical interface elements in their cars with virtual onces in form of a touch screen positioned at the center console [techradar, 2017]. It substitutes for example media or air flow/temperature controls. Microsoft is tackling the business area of group collaboration with their Surface Hub [Microsoft, 2017b] to support teamwork. Multi-touch tabletops also becoming more and more widespread [Kaltenbrunner et al., 2006, Ishii, 2008].

The Rolls-Royce future shore control center [YouTube, 2017b] is an example of a future workplace concept. It contains almost all of the previously mentioned areas of touch screens. A touch screen fixed at a workstation providing selection and drag & drop in connection with a laptop for

data input, a tablet in conjunction with a wall mounted touch screen used by operators, and a collaboration table to work together with multiple people.

Touch screens benefit from a merged in- and output area.

One obvious benefit of touch screens is the merge of the in- and output area into one. This leaves more space on smartphones for the screen itself, which enables developers to provide more information and a better overview for the user. The interaction with the UI is direct and therefore it is not necessary to translate the hand movement to a cursor at a different area like a desktop monitor.

Haptic experience and tactile feedback missing on touch screens.

Nonetheless, there are several drawbacks using a touch screen like the missing haptic experience, tactile feedback, and "issues related to discovery, exploration and learning inherent to gesture-based interaction" [Morales González et al., 2016]. For example writing without looking at the keyboard or playing/pausing/skipping music without visually searching for the controls is becoming a problem. Eyes-free interaction is not possible anymore when using a touch screen. Physical controls on the other hand can provide you with the current level/status by just touching/feeling it.

Tangible user interfaces try to tackle the drawbacks of touch screens.

Research tackled these drawbacks already in the mid-nineties by Fitzmaurice et al. [1995] with Graspable User Interfaces and Tangible User Interfaces (TUIs) by Ishii and Ullmer [1997]. TUIs "give physical form to digital information, letting serve as the representation and controls for its digital counterparts. They make digital information directly manipulatable with our hands and perceptible through our peripheral senses through its physical embodiment" [Ishii, 2008] (Figure 1.3). The physical form is "composed of both a physical handle and a virtual object" [1995] to which we refer as *tangible* throughout the thesis. Plenty of TUIs have been developed over the last decade like Illuminating Clay [Piper et al., 2002], reacTable [Kaltenbrunner et al., 2006] as well as research published proving their benefits [O'Malley et al., 2004, Tuddenham et al., 2010, Schneider et al., 2011].

Tangibles are physical objects which have a virtual representation.

Many different tangibles have been built using different kinds of tracking technologies. GaussBits and GaussBricks

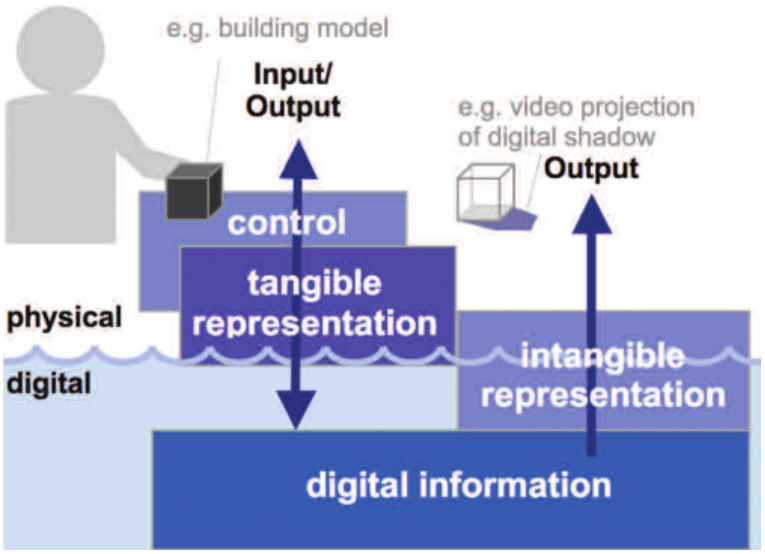

Figure 1.3: Visualization of the concept of Tangible User Interfaces. Image taken from Ishii [2008].

Systems able to detect tangibles are mostly using additional tracking technology or active tangibles.

[Liang et al., 2013, 2014] for example use an analog Hall-sensor to track magnetic structures. Both, PERCs [Voelker et al., 2015] and Actibles [East et al., 2016] are using electronic hardware internally. PERCs uses this hardware to be reliably detectable by capacitive touch screens. Actibles on the other hand incorporates a smartwatch to interact with as well as an RGB LED array arranged in a ring. Actibles are detected through an optical tracking system using fiducial markers attached to the tangibles. In 2016, an active tangible for capacitive screens entered the consumer market known as the Microsoft Surface Dial [Microsoft, 2017a] (Figure 1.4). The closest approach yet to bring back the haptic experience and tactile feedback to virtual GUI controls is the work of Weiss et al. [2009]. They built a set of tangibles called SLAP consisting of a keyboard, keypads, rotary knob, and slider for a multi-touch tabletop. Their approach uses *frustrated total internal reflection*, an optical tracking method. Markers attached to the bottom of the tangibles reflect light back to cameras below the screen. The arrangement of the markers is used as a footprint to identify a tangible. Unfortunately, the SLAP tangibles cannot be used on almost all current touch devices because capacitive tracking technologies are used. Approaches like

Figure 1.4: Image shows the Surface Dial placed on a screen which presents GUI elements around it. Taken from onmsft.com [2017].

PERCs [Voelker et al., 2015] and Microsoft (Surface Dial) are using electronic components to be detected by capacitive screens. A reason for using additional built-in electronics is the adaptive filtering mechanisms of most touch screens. Passive tangibles are filtered out after some amount of time depending on the used device [Voelker et al., 2013]. The problem with the filtering is that the underlying TUI software can not distinguish between a user lifted tangible or a tangible which was filtered out. Fortunately, this problem is less interfering than years before. The iPad Pro for example is able to detect passive tangibles for at least 25 minutes[1]. We assume that the filtering problems will further decrease or disappear entirely as the support of tangibles increases (e.g., Surface Dial). All the electronics in general make an active tangible more complex, vulnerable, complicated to maintain, and obscure the screen. A passive tangible, without the need of any electronics can be built cheaper, be made transparent [Voelker et al., 2013], and reduce maintenance.

Passive tangibles suffer from adaptive filtering algorithms.

Adaptive filtering may not be a problem in the future anymore.

The work of this thesis covers investigation of parameters (Chapter 3 “Approach”) which can be used to identify and differentiate multiple passive tangibles simultaneously on

Overview of the contents of this thesis.

[1] Test was done through a video capture of a passive tangible placed on the iPad Pro with visualized touch points.

a capacitive touch screen as well as detecting their rotation. Related work about other passive tangibles regarding used parameters and software are discussed in Chapter 2 "Related work". We use tangibles with three markers. Different parameters will be investigated like the size of the tangible and the markers. Additionally, we are looking at the internal angles of the triangle formed by the three markers. A test bench is developed to place tangible prototypes several hundred times on an iPad Pro. Data will be analyzed and evaluated in Chapter 4 "Analysis and Evaluation". Tolerance values are calculated and suggestions are made on how to use the analyzed parameters. Because the iPad Pro currently has a less aggressive filtering we developed an iOS SDK as a proof of concept which fulfills the aforementioned attributes. Detailed descriptions of the SDK as well as of each class especially how tangibles are detected, identified and compared can be found in Chapter 5 "The *PASTA* SDK". The SDK is available at GitHub[2] and as a pod at CocoaPods[3]. Limitations of the SDK due to high error values of parameters and not yet implemented features are discussed in Chapter 6.2 "Future Work".

[2]https://github.com/aroyarexs/PASTA
[3]https://cocoapods.org/pods/PASTA

Chapter 2

Related work

Using passive tangibles on capacitive screens we can only rely on the data of touches detected by the underlying system. The touches are produced by conductive markers mounted below the tangible which have a conductive connection to each other [Voelker et al., 2013].

Two major approaches of passive tangibles.

There are currently two approaches of passive tangibles which can be differentiated by the way the user has to interact with them. One where the user has to touch the tangible to be recognized by the screen (self capacitance). The other works without any user involved (mutual capacitance).

Projects using the self capacitance approach.

Self Capacitance Using the self capacitance approach tangibles have to be touched by a user to be detected by the capacitive screen. The markers attached to the bottom are wired to a conductive material on the outside of the tangible. While the user touches the tangible the markers are grounded through the user. CapWidgets [Kratz et al., 2011], CapStones & ZebraWidgets [Chan et al., 2012], and CapCodes [Götzelmann and Schneider, 2016] are some examples using this approach. TouchTokens [Morales González et al., 2016] also need the touch of the user to be detected but uses a different approach. They use tokens made out of non-conductive material with notches constraining the users' grasp. While grabbing the token, fingers are placed at the notches, which then generates touches on the screen.

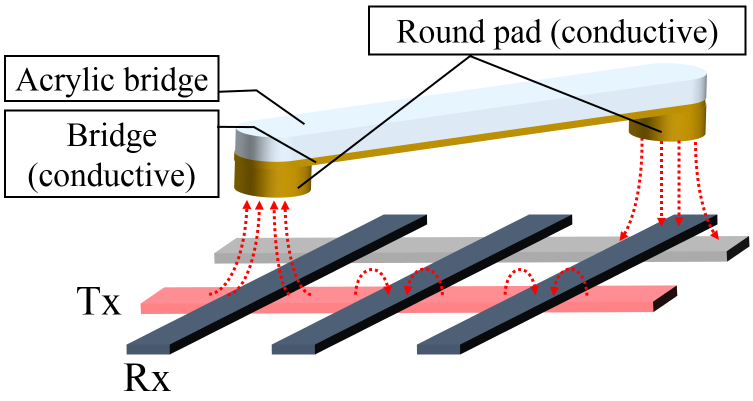

Figure 2.1: Visualization of the capacitive coupling between a marker and electrode (red arrows). Tx are transmitting and Rx are receiving electrodes. Image from Voelker et al. [2013].

Explanation of the mutual capacitance approach.

Mutual Capacitance The other approach uses a method called capacitive coupling. PUCs (**p**assive **u**ntouched **c**apacitive widgets) [Voelker et al., 2013] and TriPOD [Di Fuccio et al., 2017] are two examples for this approach. They are utilizing the grid of transmitting (Tx) and receiving (Rx) electrodes consistent in capacitive touch screens (Figure 2.1). The conductive connected markers connect currently scanned active intersection with inactive intersections creating a touch at the active intersections because the inactive one serves as ground.

Drawbacks of both approaches.

Both approaches have some drawbacks. Using the self capacitance approach tangibles are only detected while the users touch them. The system cannot distinguish between lifting and releasing a tangible. The mutual capacitance method has the problem that if two connected pads are aligned on the same set of electrodes they cannot be detected. Therefore, a tangible with at least three markers is used to always have one marker generating a touch while the other two are grounded. Additionally, touches have some noise in the input data which has to be considered in form of an error tolerance [Bottino et al., 2016, Di Fuccio et al., 2017]. Throughout this thesis we focus on mutual capacitance tangibles since we do not want the user to constantly have to touch a tangible. It would be optimal

to have the functionality of active tangibles using passive ones.

We will now discuss projects following the mutual capacitance approach. Focus is on the parameters they used to identify a tangible, rotation detection, simultaneous tangible detection and how they differentiate them.

TUIO TUIO [Kaltenbrunner et al., 2005] "is an open framework that defines a common protocol and API to transmit [...] abstract description of interactive surfaces, including touch events and tangible object states" [Kaltenbrunner, 2009] over a network. TUIO is mentioned just for completeness because many projects implement it. A complete list of projects is available at the TUIO website[1]. Basically every framework which supports TUIO should be able to work with tangible objects. But this does not mean that they are able to detect tangibles from touch input. After reviewing them it seems none except one (TouchTokens) are able to detect tangibles using a capacitive screen. TouchTokens will be reviewed later. Many of them are using computer vision to track finger input and objects, like reacTIVision [Kaltenbrunner, 2009].

A protocol used to transmit description of interactive surfaces like touch events and tangible objects.

MTK Linden [2015] developed a macOS/iOS multi-touch framework (MTK) supporting different kinds of input sources. MTK is written in Objective-C. It can recognize tangibles like PUCs and PERCs with favoring and focusing on PERCs. PERCs use additional capacitance and light sensors that data is communicated to the MTK. Thus information needs to be processed making the whole algorithm more complex. The spatial configuration of the PUC's conductive markers is used to identify them, calculating a position, and their orientation. The amount of markers is bound to three. MTK uses the distance between the markers as a pattern to identify the tangible. No further details about the recognition of patterns is given nor any error values taken as a tolerance. The internal angles, described by the triangle with the markers as vertices are used to compute the rota-

A macOS/iOS framework supporting different input sources and tangibles.

[1] https://www.tuio.org/

tion. Several algorithms are implemented to recover lost markers. But if two markers are lost and the tangible is moving away from the position where it lost the markers their new position and orientation is ignored. The framework is not publicly available. Since MTK supports different input sources it provides a lot of options such that an application to configure the framework was developed.

A Java/Android application capable of detecting one token simultaneously.

TouchTokens TouchTokens are the combination of a Java/Android based application and passive tokens [Morales González et al., 2016]. The application is able to identify tokens by using the spatial configuration of three touch points as a pattern. The difference to PUCs is that touches are generated by the user's fingers and not a conductive material. The fingers are touching the surface directly while holding a token with fingers placed at some notches of each token. Investigations of the code of their Android project [LRI, 2017] showed that only one token at a time is able to be detected. Since their recognition is based on the *best input alignment* (lowest distance between each touch) to a given pattern it will always recognize a token even if the pattern looks totally different. Also, if more than two fingers are lifted up the token is not detected anymore and no algorithms to recover lost touches are implemented. Additionally, rotation of the token is not provided. They used the centroid computed from the three touch points as position of the token. The Java (non-Android) version of the project supports TUIO.

A commercial multi-touch and TUI software.

GestureWorks GestureWorks is a commercial multi-touch and TUI software [GestureWorks, 2017]. It works with tangible objects using conductive markers. Widget objects can be configured by placing them on the screen, the system then reads in the touch points, and you provide a name at the end. No information is available whether these tangibles are operating on all capacitive screens or just on the ones which Ideum, the company behind GestureWorks, is selling with the framework.

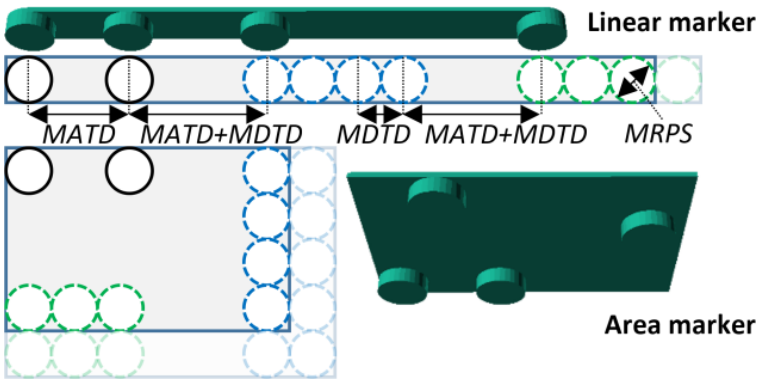

Figure 2.2: Image shows CapCodes arrangement of four markers to identify a tangible. Two black markers signaling the start of the encoding. Blue and green markers then encode the individual ID. MATD is the minimal adjacent touch-point distance (11 mm), MDTD the minimal discriminable touch-point distance (5 mm), and MRPS the minimum reliable pad size (5 mm). Image from [Götzelmann and Schneider, 2016].

CapCodes Götzelmann and Schneider [2016] developed a method to 3D print passive tangibles. The conductive material is directly printed into the object and internally connected to a point or area on the surface of the object. The user has to touch the area to generate touch points. Identification, position and rotation are gathered through four markers at the bottom of the tangible. Two of these markers are used to encode an ID by measuring the distance between them. The other two markers have to be placed at specific distances to each other and the two ID markers (Figure 2.2). They function as start sequence for the two ID markers and always have the same arrangement on the marker. Markers with 5 mm diameter are used (minimum reliable pad size (MRPS)). The minimal distance between to markers is 11 mm (minimal adjacent touch-point distance (MATD)). Minimum tangible size is 31×21 mm allowing for 12 different IDs. Linear and area tangibles are possible where linear tangibles have the four markers aligned in a row while area markers arrange them on a rectangle. No information about noise in the input, a tolerance value and about the detecting algorithm are given.

A 3D printed tangible formed by four markers. Marker arrangement features encoding of an ID.

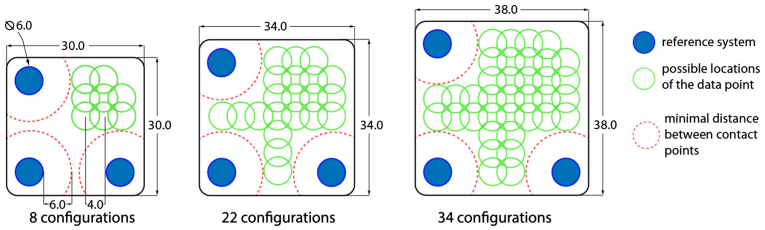

Figure 2.3: Shown is the arrangement of four markers to identify a tangible using GAINE. Three markers (blue dots) form a reference system. A fourth marker (possible positions shown with green circles) encodes the ID of the tangible. Image from Bottino et al. [2016].

GAINE is a multi-platform framework capable of detecting tangibles with four markers in a special arrangement.

GAINE GAINE [Bottino et al., 2016] is "a flexible framework for the rapid prototyping and development of edutainment and entertainment applications based on multitouch and tangible interaction". It is running on multiple platforms like Windows, Linux, macOS as well as mobile OS like Android and iOS using Unity 3D. It has built-in detection of passive tangibles using a specific arrangement of four conductive touch points. Three are used to determine position and orientation. The position of the fourth defines the ID and has to lay inside the rectangle which is defined by the three markers (Figure 2.3). Most touch panels of common commercial tablets only detect ten touch points simultaneously [Bottino et al., 2016, Di Fuccio et al., 2017]. Therefore GAINE can only detect two different tangibles and two fingers simultaneously. The software is able to distinguish 8 tangibles with a minimum size of 30 mm in width and height. The amount of distinguishable objects increases while increasing width and height in steps of 4 mm. For example with a size of 38 mm it is possible to differentiate 34 objects. They have mentioned noise in the input data and a tolerance they take into account. The only possible tolerance values provided are shown in their image (Figure 2.3). They block additional space around the three markers of 6 mm. The distance between the centers of the ID point needs to be 4 mm. If one or more touch points are lost the tangible will be removed. No recovery algorithms are implemented. They also mentioned that tangibles can be recognized contemporarily which is a hint of the adaptive filtering problem. Information whether their

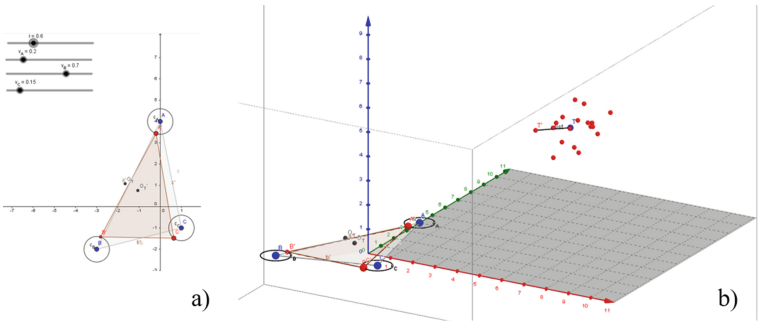

Figure 2.4: Visualization of the TriPOD approach. Blue dots indicating the actual touches. Red dots in *a)* shows one accepted configuration. Red dot cloud indicates all the accepted configurations (*b)*). Each distance between two markers is assigned to an axis in a 3D Cartesian space (*b)*). Image from [Di Fuccio et al., 2017].

project is publicly available could not be found.

TriPOD TriPOD is able to detect capacitive tangibles using three conductive markers [Di Fuccio et al., 2017]. An Android application is detecting touch-points and sends the information via WiFi to a control software running on a separate device. The control software is then detecting tangibles and sends this information back to the Android application. Currently only one tangible at a time can be recognized. Their algorithm is creating a vector in a 3D Cartesian space based on the distances between the three blue markers where each distance is assigned to one axis (blue dot in Figure 2.4 *b)*). The red dot cloud indicates all accepted configurations based on the three blue touch points. The ID of the configuration with the lowest distance to the blue dot will be assigned to the tangible. TriPOD is able to detect 24 different tangibles and it allows learning of new ones. Tangibles use markers with a size of 5 mm leaving at least 5 mm space from each other. 12 pixel error tolerance of the diameter for each marker is taken into account. To the best knowledge of the author the project is not publicly available.

A software communicating over WiFi to detect tangibles with three markers. Only one tangible at a time can be recognized.

In the next chapter we present parameters, technical setup

and a study of how we are going to solve passive tangible identification and rotation detection.

Chapter 3

Approach

GAINE [Bottino et al., 2016] and CapCodes [Götzelmann and Schneider, 2016] use tangibles with four conductive markers, enabling them to differentiate between a huge amount of patterns. But the maximum number of simultaneously detected touches is limited on a touch screen. Additionally, if two markers are aligned on the same electrode they cannot be detected anymore. Using more markers also increases the chance of aligning two markers on one electrode. Therefore, we focus on tangibles with three markers, like TriPOD [Di Fuccio et al., 2017] and PUCs [Voelker et al., 2013].

We will use tangibles with three markers.

We are targeting the iOS system because the iPad Pro has a significant decreased time after mutual capacitive tangibles are filtered out.

iOS is the target platform.

GAINE and TouchTokens already showed that it is possible to detect tangibles using an analytical approach. Both approaches map the touch inputs to the closest saved pattern. This always results in a mapping. We want to be able to filter out a tangible if it is not similar to a predefined pattern. And unlike TouchTokens and TriPOD we want to recognize multiple tangibles simultaneously. Therefore, we need a tolerance value to tackle noise in the input data to reliably detect tangibles. We will test the following parameters:

Multiple tangibles should be detected simultaneously.

- Tangible size (radius/diameter)
- Markers as vertices of a triangle:
  - interior angles
- Touch radius

Tangible size will be evaluated by calculating the circumcircle [mathopenref.com, 2017] and triangle area. To detect the rotation of a tangible we want to use the interior angles or the touch radius. Detecting an interior angle or touch radius which is not equal to all others, i.e., the smallest or the biggest, provides us with the information of the tangibles rotation.

Analysis and evaluation of the parameters is done in Chapter 4 "Analysis and Evaluation".

3.1 The Tangible Prototype

Tangibles are designed using OpenSCAD.

We describe the geometry of the tangibles in OpenSCAD[1] scripts. With this approach we can easily change parameters of prototypes without designing a new model. The script files are also available at Thingiverse[2]. A 3D sketch of the tangible components is shown in Figure 3.1. We used a lasercutter[3] to build the tangible prototypes.

Prototypes consists of a *body* and *bottom* whereby the *bottom* defines the marker arrangement.

The tangibles consists of two main parts. A *body* filled with heavy material like lead grist [Eisenmax, 2017] to create a fair amount of self-weight and a *bottom* area containing the markers. The *body* itself consists of two covers and two rings with different radii. The lead grist is filled inside the rings, between the inner and the outer one. The covers are glued on the bottom and top of the rings to form a rigid body. The *bottom* defines the arrangement of markers and provides cutouts for the tangible-Robot mount. It is made out of 5 mm perspex. The markers are placed below

[1] http://www.openscad.org/
[2] https://www.thingiverse.com/thing:1810704
[3] http://hci.rwth-aachen.de/lasercutter

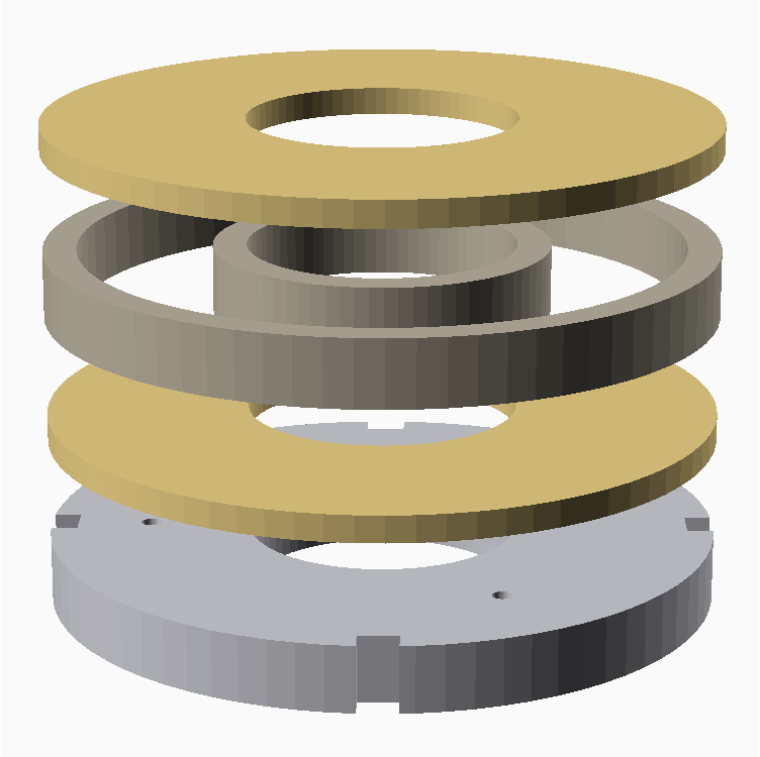

Figure 3.1: A 3D sketch of a tangible. The bottom, kept in gray, shows the cutouts at the side and the holes for the cables. The covers for the rings are shown yellowish. The rings (gray-brown) are filled with heavy material and glued to the covers.

the three holes (Figure 3.2). On the upper side we drilled three grooves from each hole guiding to the center. For each marker a cable is stripped and soldered on the marker. Then it is fiddled into the hole and placed along the groove to the center. *Body* and *bottom* are glued together then. We put an adhesive copper band on the inside of the inner ring. Last but not least we soldered the cables onto this copper band to have a stable conductivity and connection. As conductive material for the markers we used conductive shielding gaskets [we online.de, 2017b].

An overview of all build tangible prototypes is given in Table 3.1. Voelker et al. [2013] tested marker sizes of 4 - 10 mm with 2 mm steps in between and ring tangibles with radii

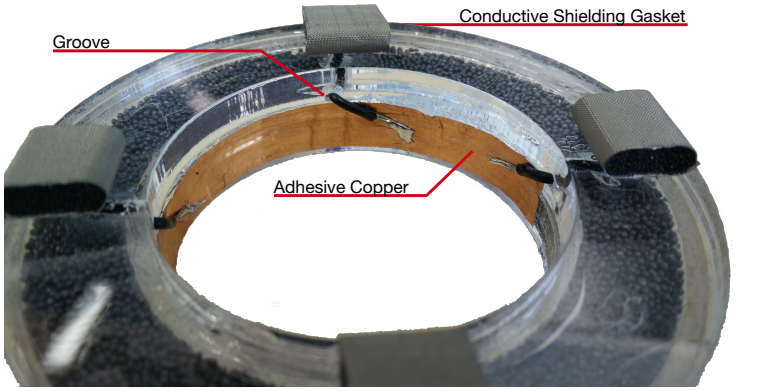

Figure 3.2: Photo of a tangible prototype. The tangible is placed bottom up. It shows the conductive shielding gasket, the adhesive copper band and the groove used as cable tunnel. Only three markers (left, right, top) are soldered to the copper band. The fourth is placed there for balance reasons.

Different tangible and marker diameter are tested.

of 20 - 50 mm. We will also try out bigger marker sizes of 13, 15, and 21 mm. Since we are using an iPad we restrict the tangible radius to a maximum of 37.5 millimeter. Bigger ones seem unsuitable because they will occlude a lot of the iPad's screen space and may feel uncomfortable to grasp.

3.2 Technical Setup

To automate the process and negating the influence of the human factor we created a test bench consisting of a robot, an iPad application, and a Java program. Each unit will be explained in more detail in the following sub sections.

The Robot is build using LEGO Mindstorms [LEGO, 2017a] (Figure 3.3). To program LEGO Mindstorms via Java we used leJOS EV3 [leJOS, 2017], a Java Virtual Machine which has to be installed/flashed to the EV3 brick via microSD card. Each component is connected with a cable to and controlled through the EV3 brick. Additionally, a WiFi

diameter (mm)	marker size (mm)	interior angle (°)
50	6	60/60/60
50	8	60/60/60
50	10	60/60/60
54	6	50/60/70
58.3	6	90/45/45
62	10	45/60/75
66.6	6	90/45/45
70	10	40/60/80
75	6	60/60/60
75	8	60/60/60
75	10	90/45/45
75	10	30/75/75
75	13	30/75/75
75	15	30/75/75
75	21	30/75/75
75	15-15-10	30/75/75
75	21-21-10	30/75/75

Table 3.1: A list of constructed tangible prototypes.

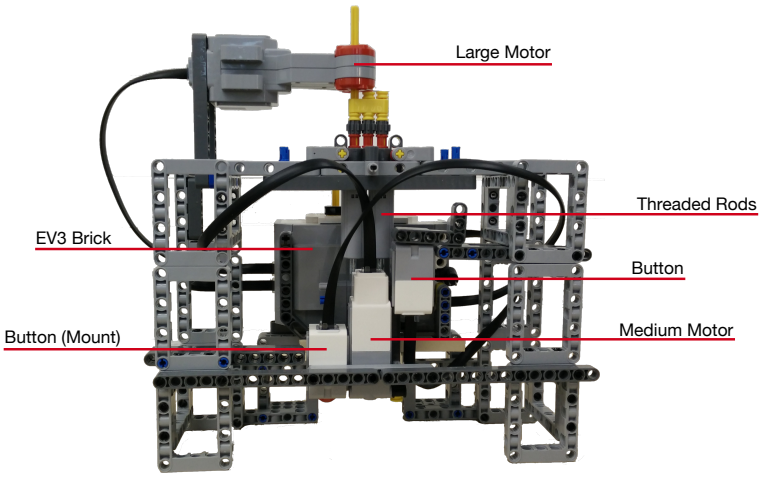

Figure 3.3: The robot used to place tangibles on the iPad. The large motor moves the module consisting of both buttons and the medium motor, up and down. It is connected to the threaded rods. The robot is controlled through the EV3 brick.

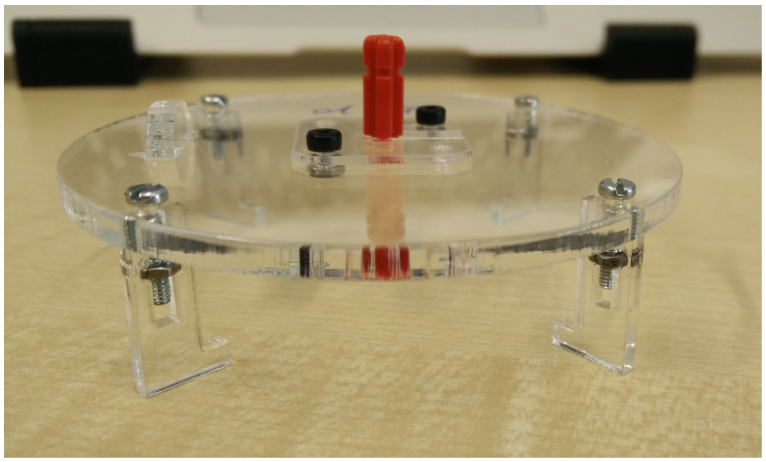

Figure 3.4: The tangible mount with 4 hooks, a LEGO cross axle rod on top (in red), and an elevation in form of a half-circle on the left side.

A robot capable of repeatedly placing a tangible on screen while rotating it.

A specifically designed mount for tangibles.

stick is attached to the USB dongle of the EV3 brick to control the robot over the network. The large motor moves the threaded rods such that the module consisting of the two buttons and the medium motor can move up and down. The right button is used as a failsafe such that the up movement stops when the button is pressed. The medium motor will rotate the tangible which is attached to it using a specifically designed mount. The rotation resolution of the medium motor is 1-degree [LEGO, 2017b]. The mount for a tangible is made out of 3 mm perspex (Figure 3.4). It has 4 hooks which fit into the cutouts of the tangible. On top is a LEGO cross axle rod fixed to the mount. It snaps into the cross hole of the medium motor. Additionally, there is an elevation, a half-circle on the upper side which is used to initialize the orientation of the mount. The left button of the robot is triggered by the elevation of the mount.

iPad Application A tangible is put down on an iPad Pro (12.9", model number: A1652) [Apple, 2017c]. We have written a small application which detects three simultaneous touches (`UITouch`) as a tangible. Only one tangible is placed at a time. If a tangible is detected a two-second-timer is started to ensure that the tangible is reliably de-

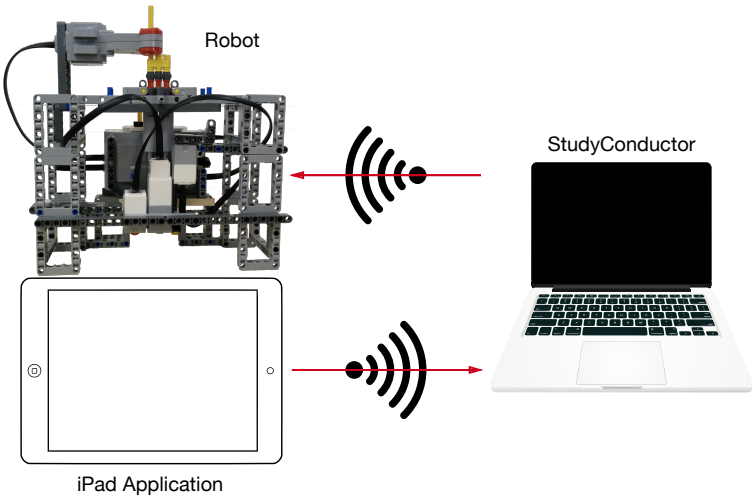

Figure 3.5: Figure shows the interaction between the robot, iPad application, and the Java program.

An iOS application transmitting touch data via HTTP.

tected and not coincidently. If the tangible is still detected within these two seconds the data is transmitted to a web server via HTTP. Additionally, every single detected touch is transmitted for completeness even if no tangible was detected.

A Java application controlling the robot and processing incoming touch data.

StudyConductor is a Java program written to control the robot in reaction to the input data of the iPad application (Figure 3.5). The program connects to the EV3 brick through the wireless network. The commands are executed on the EV3 brick using Java Remote Method Invocation (RMI) [Oracle, 2017b] with classes and methods of leJOS. The StudyConductor runs a Jetty [Eclipse, 2017] web server to handle HTTP requests from the iPad application. The received data is parsed using the JSON Processing [Oracle, 2017a] library and saved to a CSV file using opencsv [opencsv, 2017]. When a tangible is placed on the screen, the order of detected markers may vary. To map transmitted touch points and their data to already stored once we have written the `MarkerMapper` class. This class compares transmitted touch points with the previous stored and calculates the distance between them. The pair with the lowest distance results in a mapping and we continue comparing

the remaining two touches.

3.3 Study

One marker of every tangible is always in front of a cutout (see again Section 3.1). The tangible is put into the mount in a way such that the cutout with the marker is hooked into the hook left of the elevation (looking from above at the mount). *Rotation angle* and *number of full rotations* can be specified in the StudyConductor program before running it. We set rotation angle to 5 and number of full rotations to 10. This results in 720 rows of data per tangible. After StudyConductor connects to EV3 brick the initialization starts. The tangible mount is rotated clockwise such that the elevation on top of the mount presses the button. This and the initial placement of the tangible inside the mount ensures that we always have the same initial start position. Then, the robot runs on his own until he reaches the previously defined *number of full rotations*. Table 3.2 gives an overview and description of the logged data. A *placement* performs the steps specified in Algorithm 1.

Algorithm 1 `placement`

1: CANCEL(`delayedPlacement`)
2: MOVE(up)
3: ROTATE BY(*rotation angle*)
4: CLEAR($singleTouches$)
5: **if** $finishedFullRotation$ **then**
6: $\quad fullRotationCounter \leftarrow +1$
7: **end if**
8: SCHEDULE(`delayedPlacement`)
9: MOVE($down$)

The process of a *delayed placement* is explained as pseudocode in Algorithm 2.

Every single touch is transmitted to the server as well as a detected tangible. The StudyConductor process the incoming data as described in Algorithm 3.

Algorithm 2 `delayedPlacement`

1: **if** $pastSeconds >= 7$ **then**
2: LOG($singleTouches$)
3: `placement()`
4: **end if**

Algorithm 3 `incomingData`

1: **if** $singleTouchData$ **then**
2: $singleTouches \leftarrow singleTouchData$
3: **else if** $tangibleData$ **then**
4: LOG($tangibleData$)
5: SCHEDULE(`placement`)
6: **end if**

Name	Description	Format/Range
Time (*)	Time at which the data is written to log file	`HH:MM:SS`
Rotation Count (*)	Number of current running rotation	$0-9$
Robot - Rotation (*)	Current tacho count (modulo 360) of the motor which rotates the tangible	$0-359$
Robot - Rotation Button Pressed (*)	Boolean value indicating the position of the elevation of the tangible mount	`true/false`
marker$I$X	X coordinate of the Ith marker	$0-1024$ points
marker$I$Y	Y coordinate of the Ith marker	$0-1366$ points
marker$I$Radius	Size of the Ith marker as radius	points
MI - angle (radians)	Inner angle at marker I	$0-\pi$
MI - angle (degree)	Inner angle at marker I	$0°-180°$
marker$I$ $<->$ marker$J$	Distance between marker I and J	points
tangibleRadius	Measured radius of the circumcircle	points
diameter (*)	Original diameter of the tangible	millimeter
markerSize (*)	Size(s) of the used marker(s)	millimeter
interiorAngle (*)	Interior angles of the triangle	$0°-180°$
triangleArea	Calculated area of the triangle	$points^2$
id	An identifier for the tangible	$x \in \mathbb{N}$

Table 3.2: Description of data gathered during study. All items suffixed with * are always logged even if a tangible was not recognized.

Chapter 4

Analysis and Evaluation

To analyze the data JMP 13 [SAS, 2017] is used.

4.1 Data Post-Processing

We wrote a Python (3.6.1) script with pandas (0.20.2) [pandas, 2017] to post-process the logs. All log files were merged into one big CSV file. We used JMP to detect and exclude outliers. We now describe the fixes and changes made to the logs.

***Robot - Rotation* Bug** There is a bug in StudyConductor. This bug did not influence the study since we could eliminate it afterwards without influencing the result. If the *Robot - Rotation* value in the last row of a *Rotation - Count* is 360 it was not treated as the beginning of the next *Rotation - Count*. Instead, modulo operation was applied resulting in 0° without increasing *Rotation - Count*. A fix is applied by increasing the *Rotation - Count* by one for values greater than 0. See Table 4.1 for an example.

Assigning Additional Columns Some of the first log files did not contain the columns *diameter*, *markerSize*, and *inte-*

Time	Rotation - Count	Robot - Rotation	...
...	...	...	...
...	1	355	...
...	1	0	...
...	2	5	...
...	...	...	...

Table 4.1: Table shows the bug which occurred during logging tangible data. The wrong *Rotation - Count* value is marked in red. It will be replaced with the next higher value.

riorAngles (see again Table 3.2). These information are retrieved from the log file name and added to the log.

The area of the triangle, with each marker being a vertex of the triangle, is computed using Heron's formula [Programiz, 2017] and assigned to the log as column *triangleArea*.

We added a last column containing IDs to separate the tangibles from each other during analysis. The ID is composed of diameter, marker size, and interior angles. The resulting ID string looks like this: *d75 6mm 60-60-60*.

4.1.1 Outliers Detection and Removal

We created a box plot of the triangle area of each tangible to detect possible outliers (Figure 4.1). We define an outlier as a data point which lays outside the range of $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$ (**Inter**quartile **r**ange). JMP visualizes selected rows in the data table immediately in every graph which references this data table by graying out all the other data points. Additionally, JMP allows selection of rows directly through the graph. Using this feature, we plot the x- and y-coordinates of each marker in separate graphs. Figure 4.2 shows the x- and y-coordinates of *marker1*. Selecting an outlier in the triangle area graph now highlights the appropriate coordinate of each marker. Since we are using three markers with a predefined arrangement we can

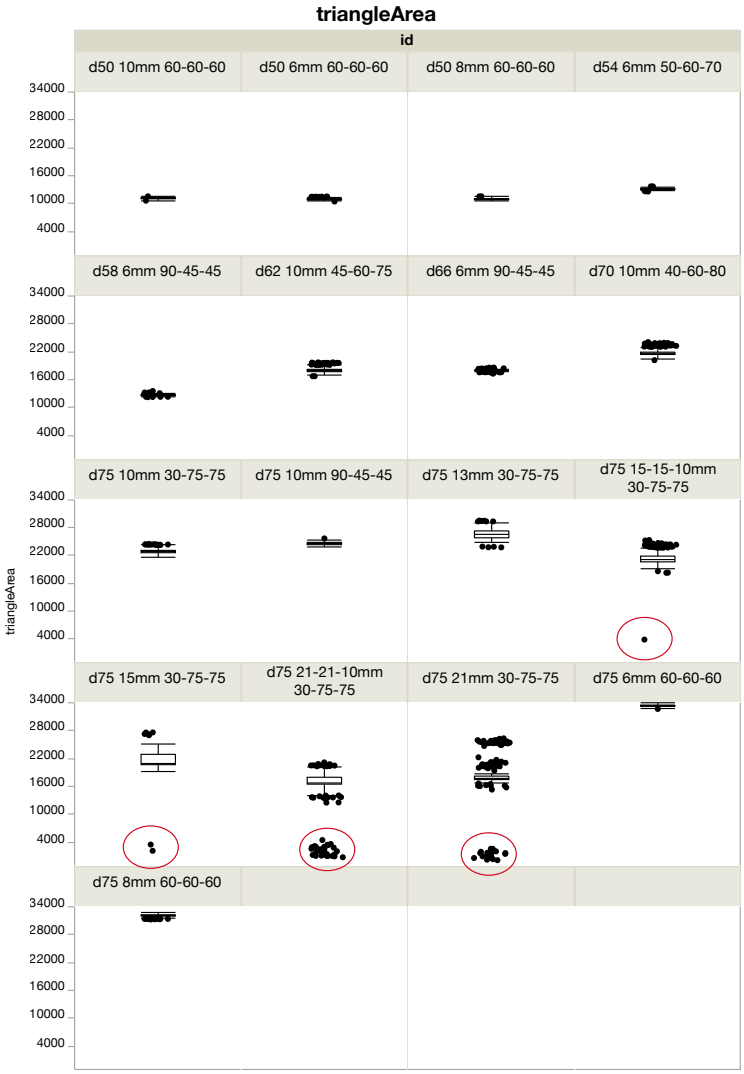

Figure 4.1: Y-axis shows the triangle area in points. Tangibles are grouped by their identifier. Whiskers of box plots representing $1.5 \times IQR$ added to $Q3$ and subtracted from $Q1$. The actually removed outliers are framed in red.

exclude rows where at least two markers seem to lay at the same angle of the circle. Figure 4.3 provides a visualization of an outlier selection. *Marker1* and *marker2* are positioned at the same quadrant of the circle. This also creates the assumption that bigger marker sizes are sometimes too big to be recognized as a single touch. We detected 50 outliers ful-

50 outliers have been excluded.

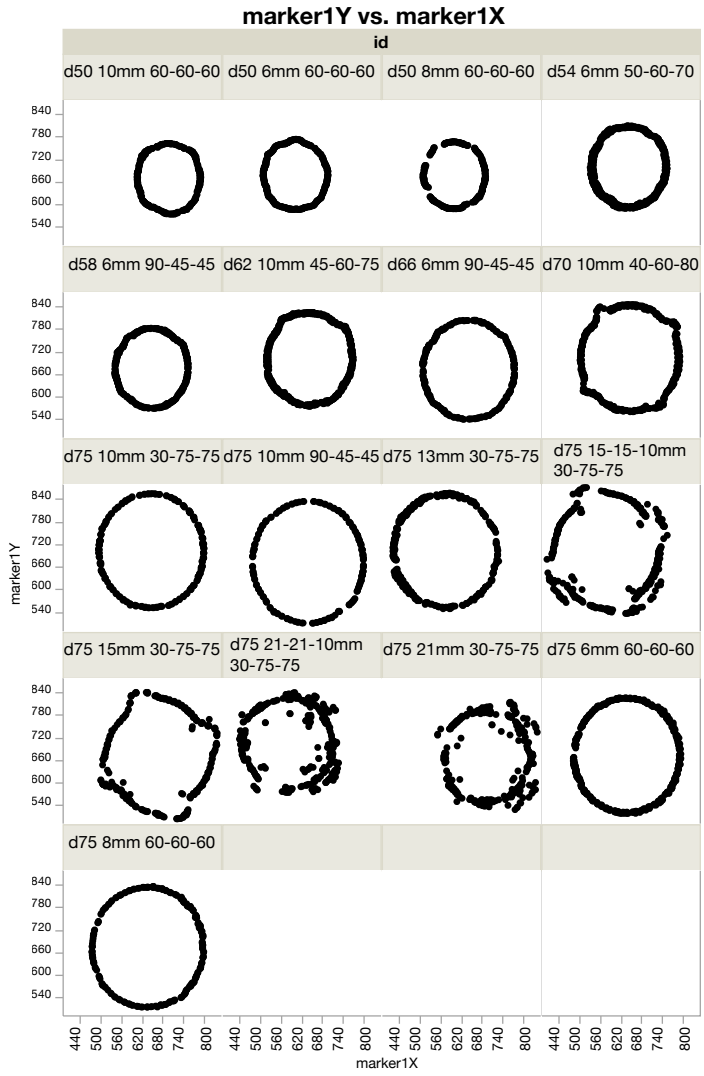

Figure 4.2: X and Y coordinates of marker 1 visualized for each tangible. Coordinates are given in points.

filling the criteria and excluded them from further analysis. They are framed in red in Figure 4.1.

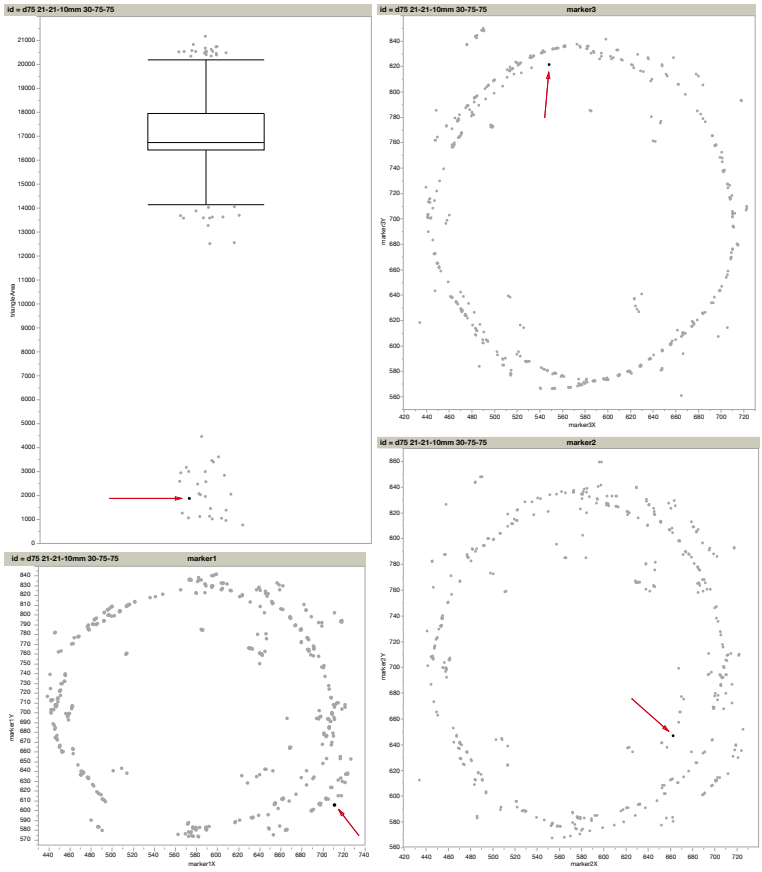

Figure 4.3: Figure shows four graphs of tangible *d75 21-21-10mm 30-75-75*. Namely *triangleArea* and x and y coordinates of markers one to three. A specific data point is highlighted in each graph which shares the same row in the data table. For better visibility a red arrow is highlighting the selected data point.

4.2 Marker Size

Figure 4.4 shows the radii gathered using `UITouch .majorRadius`. They are grouped by the size of the obscured markers. As we see right away the values are discrete and not continuous. We created a mosaic plot (Figure 4.5) showing the count of marker radii values grouped by *markerSize*. Minimum is at 10.42 and maximum at 72.97. The value of a step between two values is 10.42. The 6

Touch radius values are discrete.

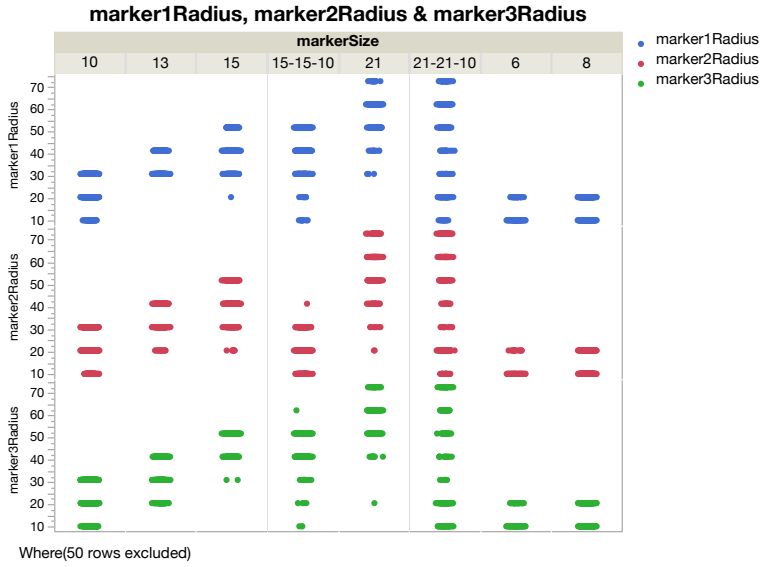

Figure 4.4: A scatter plot showing measured marker radii grouped by the obscured marker sizes. Marker radius is given in points.

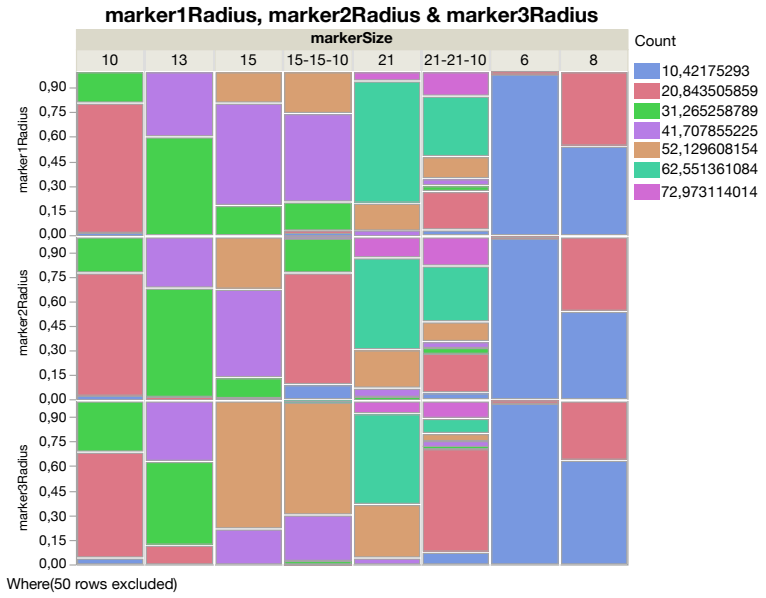

Figure 4.5: Mosaic plot showing count of marker radii values grouped by *markerSize*. Measured radii are shown right as legend with color marks.

mm marker seems very stable being recognized almost always with 10.42 points and just a few times as 20.84. This is a very promising result. Marker size with 8 mm is detected with a radius of 10.42 points in approximately two-thirds of the cases and one-third as 20.84. The overlap with the 6 mm marker is too large to distinguish these markers reliably. Better results can be achieved when distinguishing marker sizes 6 and 10 mm. The count of overlapping values is extremely small and marker size was detected as 20.84 in $70\% - 80\%$ of the cases. Therefore, in most cases it should be possible to distinguish them. The most frequent detected radius of the 13 mm marker is 31.26. Compared to the 10 mm marker the frequency is smaller than the most frequent detected radius of the 10 mm marker. Also, the influence of a third radius is slightly increasing. Nonetheless, it could be possible to distinguish 6, 10, and 13 mm markers with a smart implementation. 15 mm marker's most frequent radii is 41.70 but the frequency dropped compared to the previous marker sizes and frequencies of other radii increased as well. Additionally, one marker was mainly detected with radius 52.12. The reason for this is currently unclear. Maybe the marker was a little bit folded or tilted. The same applies to the markers of the 15-15-10 tangible except for the 10 mm marker. The 10 mm marker was detected reliably. The 21 mm marker on the other hand seems to not have the error of detecting two markers with the same size differently. The most frequent detected radii for 21 mm marker is 62.55. The frequency of value 62.55 is dropping for the 21-21-10 mm marker arrangement. In general it seems like the accuracy is dropping with rising marker size.

Since iOS provides discrete radius values we suggest to implement a 'counter' class which chooses the most frequent radius over the lifetime of a marker. Additionally, markers with 8 mm size should be avoided. Instead, markers with sizes 6, 10, and 13 should be used. Systems which provide continuous touch radius data should take an error into account.

8 mm markers should be avoided.

If a tangible is placed on the screen, the radius value of a touch will increase from the minimum value of 10.42 to the actual size. Comparing two tangibles immediately af-

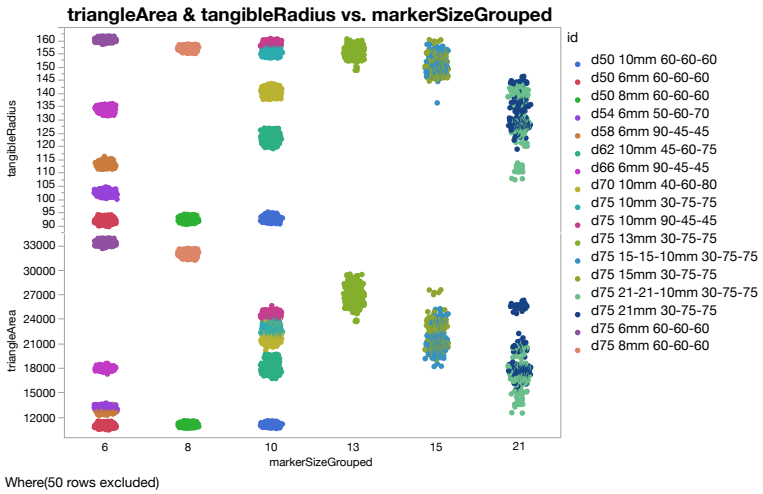

Figure 4.6: Scatter plot showing *triangleArea* and *tangibleRadius* versus *markerSizeGrouped*. Scatters for each tangible are separated by color.

ter one was detected the touch radius may not represent the actual size. Developers should keep that in mind if they want to include the marker size as a parameter.

4.2.1 Influence on Tangible Size

Looking at Figure 4.2 we clearly see that the positions of *marker1* deviate from a regular circle shape if a marker size greater than or equal to 15 mm is used. Figure 4.1 also supports this since extreme outliers only occur for these marker sizes. In Figure 4.4 we see that the range of markers with size *15*, *15-15-10*, *21*, and *21-21-10* is min 31.26 and max 72.97 points compared to just min 10.42 and max 20.84 points for the rest. We added a new column *markerSizeGrouped* which groups tangibles having mainly the same marker sizes. For example tangible with marker sizes 15-15-10 mm is grouped with tangible that has only markers with size of 15 mm. We created a scatter plot showing *triangleArea* and *tangibleRadius* on y-axis and *markerSizeGrouped* on the x-axis (Figure 4.6). As one can see the deviation of data points increases with increasing marker size. To provide further evidence we calculated the standard devi-

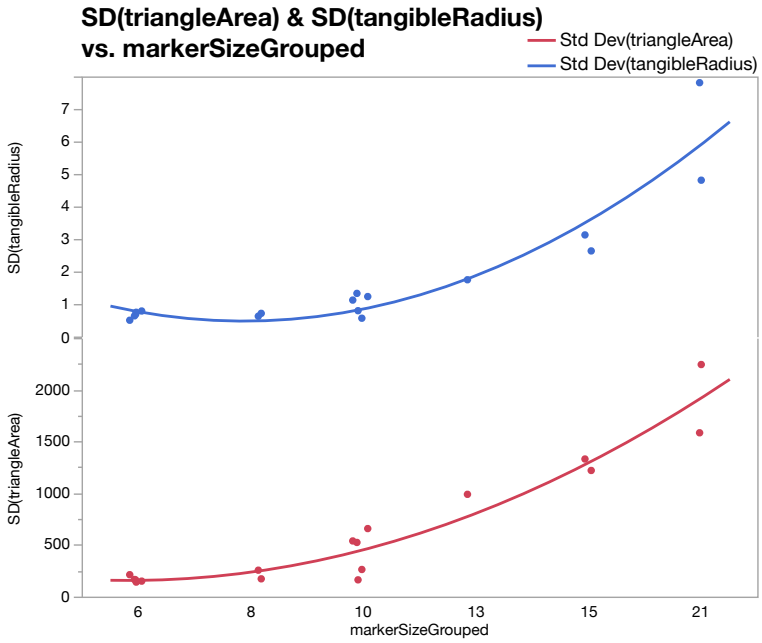

Figure 4.7: Scatter plot with a line of fit showing SD of *triangleArea* and *tangibleRadius* for each tangible versus *markerSizeGrouped*.

ation (SD) of *triangleArea* and *tangibleRadius* for each triangle. Then, we created a scatter plot showing the SD versus the grouped marker sizes (Figure 4.7). The plot shows that increasing the marker size influences *triangleArea* and *tangibleRadius* by increasing their SD. Therefore we decided to exclude tangibles with marker sizes greater or equal 15 mm from further analysis. We also recommend using marker sizes smaller than 15 mm.

Marker size influenced tangible size. Marker sizes ≥ 15 mm are excluded.

4.3 Tangible Size

Looking again at Figure 4.6 we see overlapping areas of *triangleArea* for different diameters. Comparing it with *tangibleRadius* the same areas did not overlap. Looking at Figure 4.8 we see that means of *triangleArea* area overlapping with ranges of other means (diameters 54 and 58) or that the means are almost equal (diameters 62 and 66). The rea-

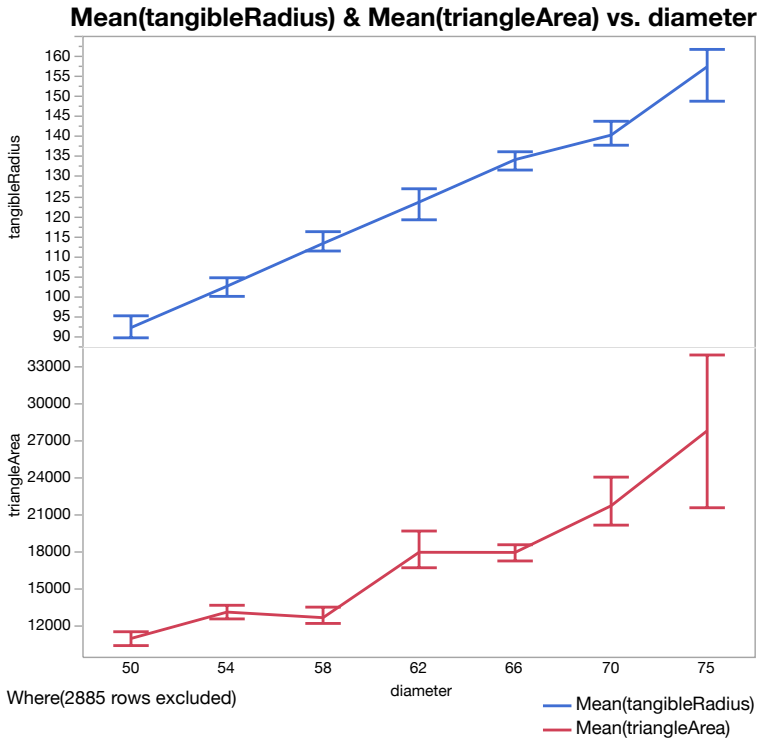

Figure 4.8: Mean of *tangibleRadius* and *triangleArea* versus diameter of build tangibles. Error bars showing the range of values.

tangibleRadius is chosen as parameter for the size.

son becomes clear because while building the tangible prototypes we arranged the markers on the circle line which already states how we expected the size of a tangible to be; namely the radius of this circle. We changed marker positions on this circle line but assumed to not change the tangible size. This influenced the triangle in two ways: it changed the distance between markers and the internal angles. By coincidence it happened that tangible *d62 10mm 45-60-75* and *d66 6mm 90-45-45* have a similar triangle area. The latter tangible describes a bigger circumscribed circle (*d*66) but looks more compressed due to the internal angles (90-45-45). The same applies for tangibles *d54 6mm 50-60-70* and *d58 6mm 90-45-45*. This makes the decision choosing *tangibleRadius* as the size parameter obvious. Figure 4.8 shows that *tangibleRadius* is almost monotonously increasing with increasing diameter. Even the range seems not to

overlap at all.

To be able to distinguish tangibles by size we need an allowed error. We want to apply the *68-95-99.7 rule* or *three sigma rule* which says that 99.7% of the distribution's values lie within three standard deviations of the mean. But the rule is only applicable if the sample comes from a normal distribution. Therefore, we will test our data whether *tangibleRadius* comes from a normal distribution.

We want to use *three sigma rule*.

4.3.1 Is Tangible Radius Normal Distributed?

Ghasemi and Zahediasl [2012] said that it "is preferable that normality be assessed both visually and throughout normality tests, of which Shapiro-Wilk test [...] is highly recommended". The H_0 hypothesis of Shapiro-Wilk and Kolmogorov–Smirnov states that the data was drawn from a normal distribution. Small p values (< 0.05) reject the hypothesis. Table 4.2 shows the results of the normality tests[1].

diameter	N	Mean	SD	Skew	Kurtosis	Normality test	
						Test	p
50	1906	92.32	0.83	0.435	0.13	$W = 0.987$	$< .0001$
54	709	102.65	102.65	-0.356	0.58	$W = 0.99$	0.0002
58	556	113.33	0.77	0.1055	0.068	$W = 0.996$	0.141
62	721	123.65	1.35	-0.17	0.377	$W = 0.992$	0.0005
66	591	134.25	0.807	-0.1805	-0.22	$W = 0.9845$	$<, 0001$
70	699	140.37	1.25	0.2785	-0.5776	$W = 0.985$	$<, 0001$
75	3374	157.44	2.27	-0.25	-0.787	$D = 0.062$	0.01

Table 4.2: Table shows row count (N), mean, standard deviation (SD), skew, kurtosis, and normality test results of *tangibleRadius* for all diameters. W is the Shapiro-Wilk test statistic and D the Kolmogorov–Smirnov test statistic.

Except diameter 58, the others are not drawn from a normal distribution and therefore H_0 is rejected. But Field [2009] said that the test has its "limitations because with large sample size it is very easy to get significant results from

[1] JMP uses Kolmogorov–Smirnov test for sample sizes greater 2000 which implies they are using the extension of Royston [1992] for the Shapiro-Wilk test. With that extension Shapiro-Wilk can be used for sample sizes of $4 \leq n \leq 2000$.

small deviations from normality". For "a large sample (200 or more) it is more important to look at the shape of the distribution visually and to look at the value of the skewness and kurtosis statistics rather than calculate their significance." [Field, 2009]. Figure 4.9 shows a histogram overlayed with fitting normal function, a box plot, and quantile plot for *tangibleRadius* grouped by *diameter*. Just looking at the histograms we assume that *tangibleRadius* comes from a normal distribution, except for diameters 66 and 75. Diameter 66 may tend a little bit to a bimodal distribution, but the quantile plot promises that it also comes from a normal distribution. Only diameter 75 is a little bit of the track. Data is a somewhat tailed and kurtosis has a higher offset to 0 with -0.787 compared to the other diameters. But we still claim that the data tends to be normal distributed.

We assume *tangibleRadius* comes from a normal distribution.

To sum up, we assume the data comes from a normal distribution. Thereby we can apply the three sigma rule. [2]

4.3.2 Determine Tangible Radius Error

Table 4.2 shows the standard deviations of *tangibleRadius* by *diameter*. The mean of the standard deviations is 1.13 points. Applying three sigma rule results in an error of 3.398 points which we have to take into account. The error has to be added to and subtracted from the radius. In other words the radii of two tangibles have to differ at least $2 \times 3.398 = 6.7966$ points (iOS) to be distinguishable because their error bounds should not overlap.

Tangible radius error is 3.398 points.

Standard deviation of diameter 75 stands out of the other values. One reason could be the 13 mm marker size. In Figure 4.6 we see that the data points of this tangible are more scattered than most of the tangibles with marker sizes lower 13 mm.

If we exclude the 13 mm marker size tangible from analysis

[2] If you come to the conclusion that our data does not come from a normal distribution and therefore three sigma rule can not be applied keep in mind that we can still apply *Vysochanskij–Petunin inequality* (or *Chebyshev's inaquality*) which means that 95% (or 89%) of the distribution's values are within three standard deviations of the mean.

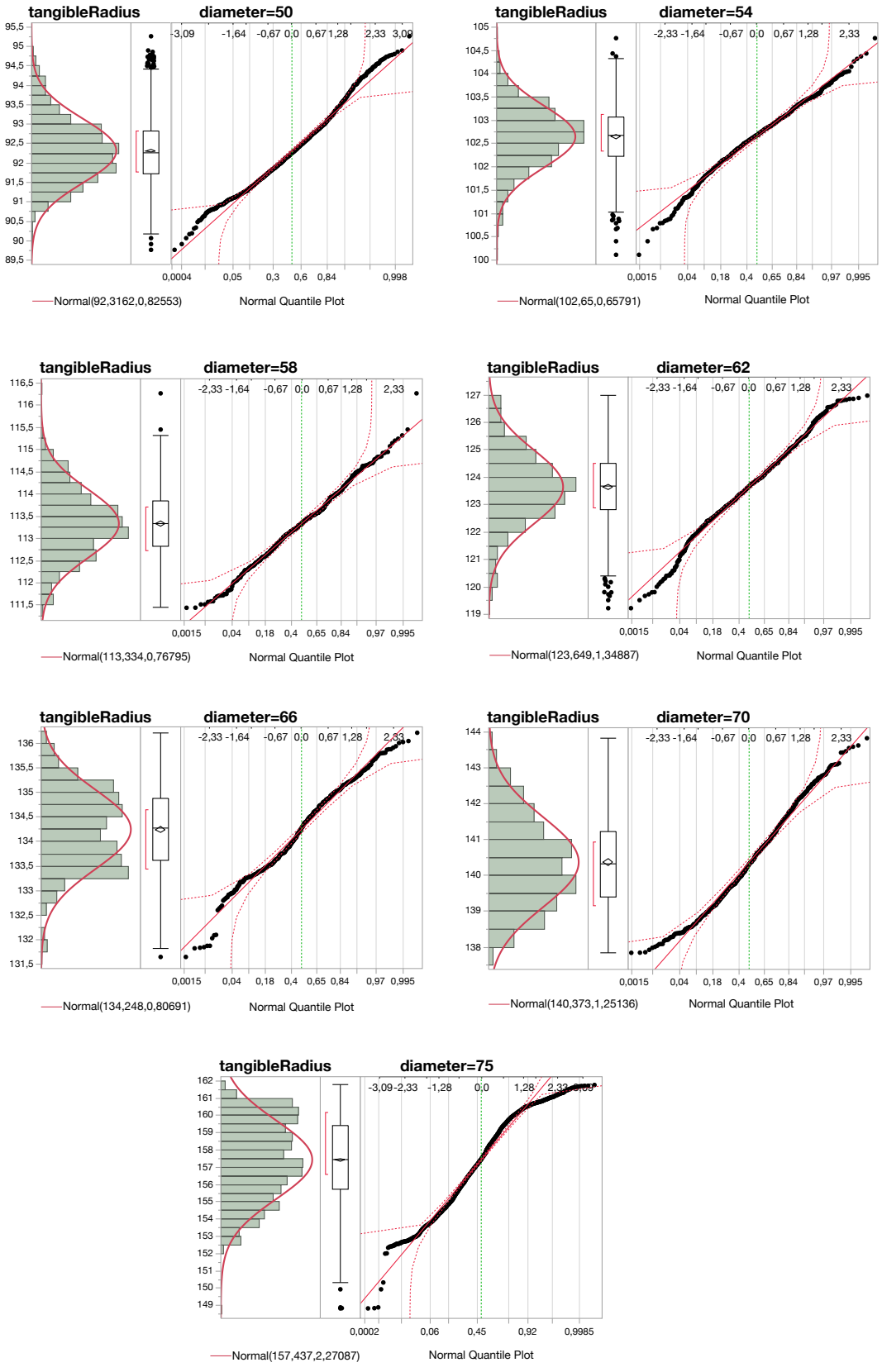

Figure 4.9: Histogram overlayed with fitting normal distribution function, a box plot, and a quantile plot of *tangibleRadius* by *diameter*.

Excluding 13 mm markers is unnecessary.

we receive a standard deviation of 1.93 points which results in a mean of 1.08 points of the standard deviations. Compared to the previous mean of 1.13 points we see that the influence of 13 mm marker size is about 0.05 points. It is unnecessary to not allow tangibles with 13 mm marker sizes to just decrease standard deviation mean by 0.05 points.

Error Converted to Centimeters

Regarding the iOS documentation of property `scale` of class `UIScreen` the scale factor is 2.0 for retina displays like the iPad Pro 12.9″ that we are using. The logical coordinate given by iOS has to be multiplied with the scale factor to get pixels. To convert pixel to inch we have to divide the measured pixels by the pixels per inch (ppi) of the device. The ppi of the iPad Pro [Apple, 2017d] are 264. Last but not least we have to convert inch to cm where one inch is 2.54 centimeters. The equation is then

$$\frac{x * 2.0}{264} * 2.54 = y \tag{4.1}$$

Tangible radius error in millimeters is 0.065.

where x are the points and y the resulting centimeters. In our case we have a radius error of 3.398 points (Section 4.3.2) which are 0.065 centimeters (0.65 millimeters).

To get a feeling of the impact of this value we calculate the number of distinguishable tangibles in the range of 50 to 75 millimeters (Table 3.1. Within this range we tested 7 different diameters. Dividing the range of 25 millimeters ($75-50$) by the number of tangibles (7) we receive 3.57 millimeters which is the average distance between the different diameters. Our calculated error multiplied by two gives us the allowed error between two tangibles namely 1.3 millimeters. Dividing the range of 25 millimeters by the diameter error of 1.3 we receive 19.23. In other words, we are able to distinguish 19 tangibles within this range.

19 tangibles can be distinguished within 50 to 75 mm diameter.

4.4 Interior Angle

The interior angles can be used for both identification and to detect the rotation of a tangible. In case of detecting the rotation a precondition is that the interior angles must not be symmetrical.

To be able to distinguish different angles we need to know again the error of the measured interior angles.

4.4.1 Testing for Normal Distribution

interiorAngles	N	Mean	SD	Skew	Kurtosis	Normality test	
						Statistic	p
30-75-75	1428	34.687	4.414	0.15	-1.69	W=0.84	$< .0001$
40-60-80	699	78.93549	2.69	-1.76	3.044	W=0.796	$< .0001$
45-60-75	721	60.29	2.1245	0.034	-0.498	W=0.976	$< .0001$
50-60-70	709	59.946	1.26	-0.198	-0.61	W=0.9865	$< .0001$
60-60-60	3213	59.9356	1.97	-0.4766	0.13	D=0.066	< 0.01
90-45-45	1786	45.31	1.10	-0.244	0.089	W=0.986	$< .0001$

Table 4.3: Table shows row count (N), mean, standard deviation (SD), skew, kurtosis, and normality test results of *M1 - angle (degree)* by *interiorAngles*. W is the Shapiro-Wilk test statistic and D the Kolmogorov–Smirnov test statistic.

interiorAngles	N	Mean	SD	Skew	Kurtosis	Normality test	
						Statistic	p
30-75-75	1428	70.95	4.325	-0.24	-0.233	W=0.96	$< .0001$
40-60-80	699	60.83	2.154	0.8359	0.87	W=0.945	$< .0001$
45-60-75	721	74.9998	2.755	-1.386	1.235	W=0.841	$< .0001$
50-60-70	709	50.397	1.096	0.126	-0.21	W=0.997	0.2226
60-60-60	3213	60.75	1.82	0.365	-0.25	D=0.095	< 0.01
90-45-45	1786	43.967	1.0278	0.1045	-0.0327	W=0.9956	$< .0001$

Table 4.4: Table shows row count (N), mean, standard deviation (SD), skew, kurtosis, and normality test results of *M2 - angle (degree)* by *interiorAngles*. W is the Shapiro-Wilk test statistic and D the Kolmogorov–Smirnov test statistic.

Table 4.3, Table 4.4, and Table 4.5 are showing some summary statistics plus results of normality test as well as skew and kurtosis of measured angles for each marker. All ex-

interiorAngles	N	Mean	SD	Skew	Kurtosis	Normality test	
						Statistic	p
30-75-75	1428	74.36	3.378	-0.856	0.626	W=0.9356	$< .0001$
40-60-80	699	40.236	1.23	0.246	0.21	W=0.99	$< .0001$
45-60-75	721	44.71	1.48	0.537	0.2279	W=0.9778	$< .0001$
50-60-70	709	69.657	1.485	-0.7996	0.3232	W=0.946	$< .0001$
60-60-60	3213	59.32	1.91	-0.99	1.8587	D=0.09	< 0.01
90-45-45	1786	90.72	0.944	-0.012	-0.41	W=0.995	$< .0001$

Table 4.5: Table shows row count (N), mean, standard deviation (SD), skew, kurtosis, and normality test results of *M3 - angle (degree)* by *interiorAngles*. W is the Shapiro-Wilk test statistic and D the Kolmogorov–Smirnov test statistic.

cept one normality test result (*M2 - angle (degree)*, *interiorAngles*: 50-60-70, $p = 0.2226$) reject that our data is drawn from a normal distribution. Since we have a huge sample size we will evaluate it visually again. Figure 4.10 shows histogram, box plot, and quantile plot for each marker angle grouped by *interiorAngles*. Interior angles of group 30-75-75 are definitely not drawn from a normal distribution. The histogram shows a bimodal distribution for all markers. We will have a detailed look into that in Section 4.4.1. Groups 40-60-80 and 45-60-75 looking a little better but still not normal distributed except marker three in both groups. Skew and kurtosis of marker one in group 40-60-80 (-1.76 and 3.044) and of marker two in both groups (0.8359, 0.87 and -1.386, 1.235) are quite high. The markers in groups 50-60-70 and 60-60-60 tend to be normal distributed except one marker of each group. In the case of group 50-60-70 this is marker three which also has a little higher skew of -0.7996 than the rest. For Group 60-60-60 it is also marker three where both skew (-0.99) and kurtosis (1.8587) are quite high. Results for group 90-45-45 look promising and therefore we assume they are drawn from a normal distribution.

Detailed Look Into Group 30-75-75

Table 4.6 shows a summary statistic, skew and kurtosis, and results of Shapiro-Wilk test. Looking at the histogram in Figure 4.11 and the calculated mean values in Table 4.6

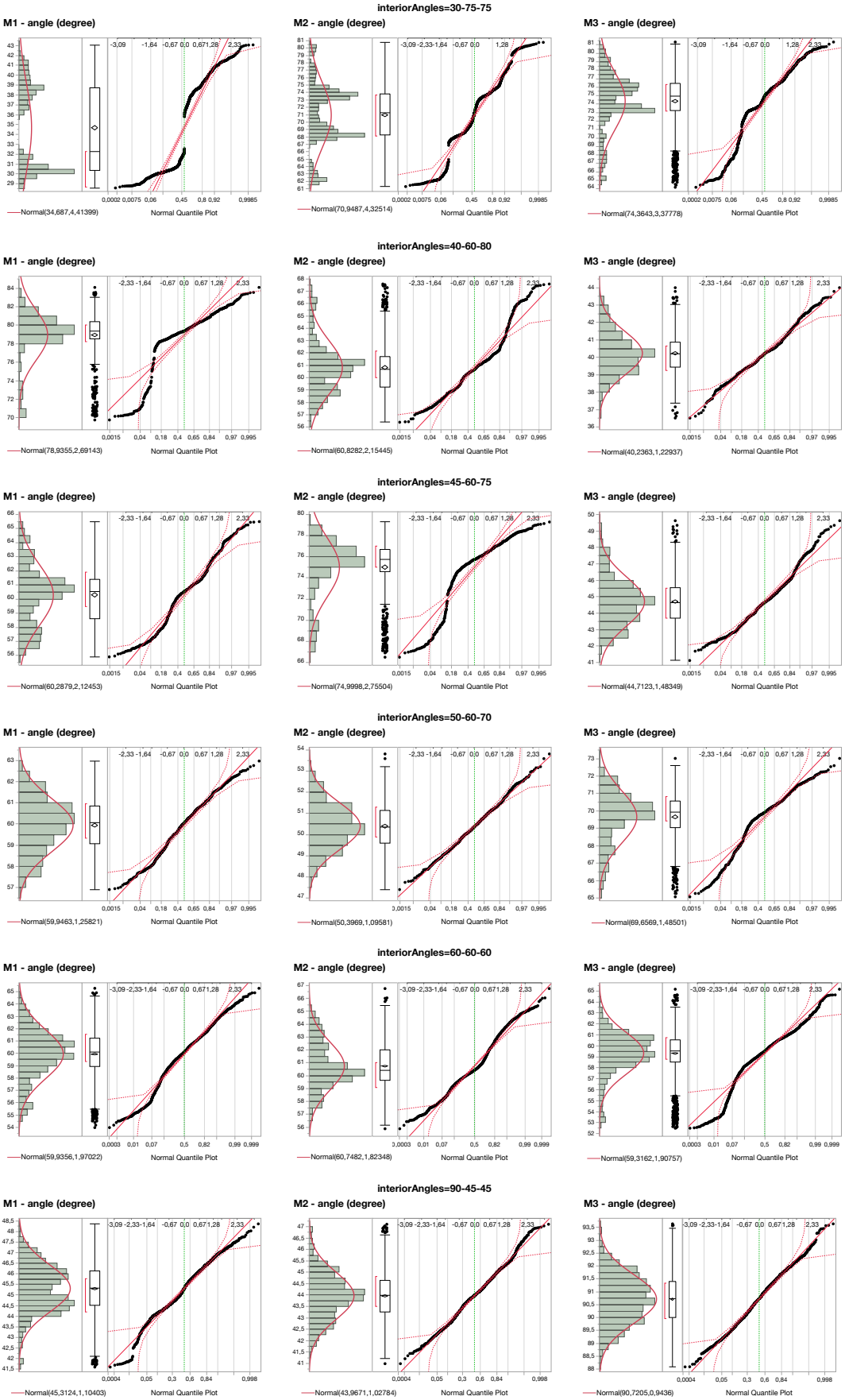

Figure 4.10: Histogram overlayed with fitting normal distribution function, a box plot, and a quantile plot of marker angles by *interiorAngles*.

id=d75 10mm 30-75-75

marker	N	Mean	SD	Skew	Kurtosis	Normality test	
						W	p
1	719	30.47	0.75	0.27	0.197	0.96	$< .0001$
2	719	73.916	2.89	0.062	0.24	0.95	$< .0001$
3	719	75.61	2.83	-0.85	0.767	0.91	$< .0001$

id=d75 13mm 30-75-75

marker	N	Mean	SD	Skew	Kurtosis	Normality test	
						W	p
1	709	38.96	1.53	0.28	-0.55	0.98	$< .0001$
2	709	67.94	3.3466	-0.38	-0.498	0.915	< 0.0001
3	709	73.0977	3.416	-0.83	0.26	0.882	$< .0001$

Table 4.6: Table shows sample size (N), mean, standard deviation (SD), skew, kurtosis, and normality test results of marker angles for *d75 10mm 30-75-75* and *d75 13mm 30-75-75*. W is the Shapiro-Wilk test statistic.

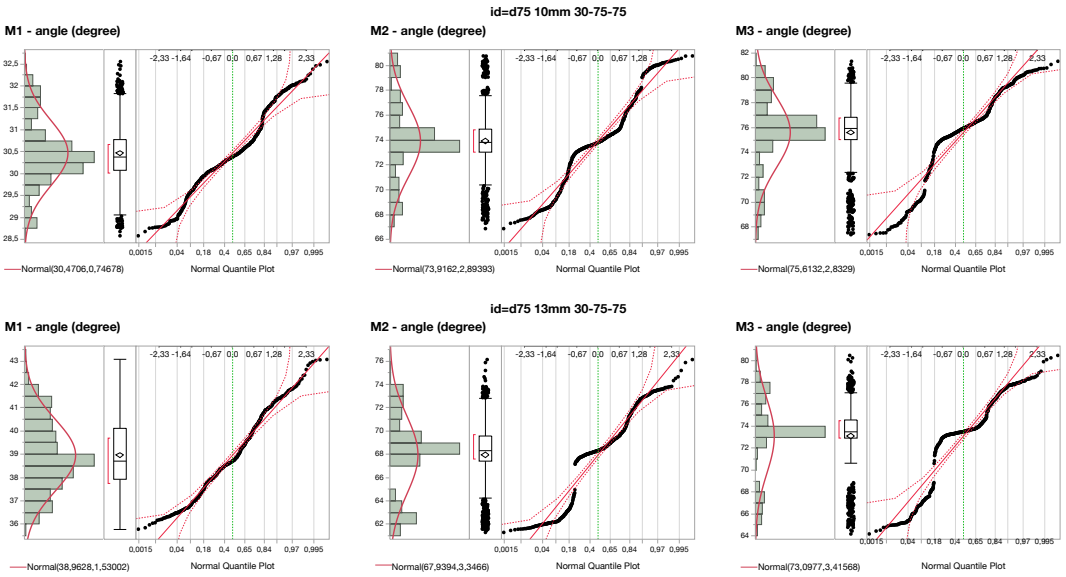

Figure 4.11: Histogram overlayed with fitting normal distribution function, a box plot, and a quantile plot of marker angles of tangible *d75 10mm 30-75-75* and *d75 13mm 30-75-75*.

compared with the values of the expected interior angles of 30°, 75°, and 75° we see that the tangible with 13 mm marker size diverges. Means of the detected angles of the 13 mm marker size tangible are 38.96, 67.94, and 73.0977. Means of the 10 mm marker size tangible are way closer to

the expected values. Also standard deviation is higher for 13 mm. This definitely explains the bimodal distribution in Figure 4.10.

Looking at both tangibles separately does not change the assumption that they are not drawn from a normal distribution. Shapiro-Wilk test results rejecting H_0 and looking at the quantile plots (Figure 4.11) mostly all markers are not describing a line which fits the normal.

Almost all samples of interior angles are not drawn from a normal distribution.

Since we cannot say that all samples are drawn from a normal distribution we cannot apply the three sigma rule and therefore have to stick to *Chebyshev's inequality* which states that at least $1 - 1/k^2$ of the distribution's values are within k standard deviations of the mean.

4.4.2 Determining the Error

We decided to split group 30-75-75 into the two containing tangibles (Section 4.4.1). To compute the error we then use the standard deviation of tangibles *d75 10mm 30-75-75* and *d75 13mm 30-75-75* itself together with the standard deviations of the other groups. We calculated the mean of these standard deviations resulting in an error of 2.066 degrees. Since our data is not normal distributed nor every sample unimodal we can not apply three sigma rule or *Vysochanskij–Petunin inequality* and therefore have to stick to Chebyshev's inequality. So taking three standard deviations into account results in 89% of the values lying around the mean. Multiplying the error by factor 3 results in 6.198 degrees. This means that two angles have to differ by $12.396°$ to be distinguishable.

Interior angle error is 6.198 degrees.

The error is suitable to detect the rotation of a tangible for specific interior angles but limits the amount of different angle patterns.

Another problem is that we only cover 89% of the samples because the data is not drawn from a normal distribution. In 6.2 "Future Work" we present another idea how to minimize the the error of the interior angle.

Chapter 5

The *PASTA* SDK

PASTA[1] is an iOS SDK developed to detect passive tangibles. It is is written in Swift 3 using Xcode 9 and is available under the MIT license. It features identification, orientation detection and recovery on marker loss.

In this chapter the PASTA SDK will be explained in detail.

5.1 Getting Started

The project is publicly available on GitHub[2] and as CocoaPods[3].

5.1.1 Installation

To install it, simply add the following line to your Podfile: `pod 'PASTA'`.

[1] Abbreviation for **PAS**sive **TA**ngible.
[2] https://github.com/aroyarexs/PASTA
[3] https://cocoapods.org/pods/PASTA

5.1.2 How to Use

1. `import PASTA`
2. Add an instance of `PASTAView` as a subview or set it as type of a view in Interface Builder.
3. Implement `TangibleEvent` and assign the object to `tangibleDelegate` property of `PASTAView`.
4. Provide some patterns by calling `whitelist(pattern:identifier:)` on `PASTAView.manager`
   (a) Create `PASTAPattern` with `MarkerSnapshot`s from scratch, or
   (b) disable whitelist by setting `patternWhitelistDisabled` to `true`, place your desired tangibles on the screen, whitelist the patterns, and enable whitelist again.

Example An example project can be found in the *Example* directory of the PASTA project. To run the example project, clone the repo, and run `pod install` from the *Example* directory first.

5.2 Class Documentation

The whole code is documented and a HTML based documentation is available online at cocoadocs.org[4]. In this section details of specific functions and properties of each class are explained in detail. A class inheritance diagram is shown in Figure 5.1. A flow chart showing the process from a `UITouch` event to a working `PASTATangible` is shown in Figure 5.2. The Metron [Heuvelmans, 2017] library is used to simplify complex 2D geometric calculations.

[4] http://cocoadocs.org/docsets/PASTA

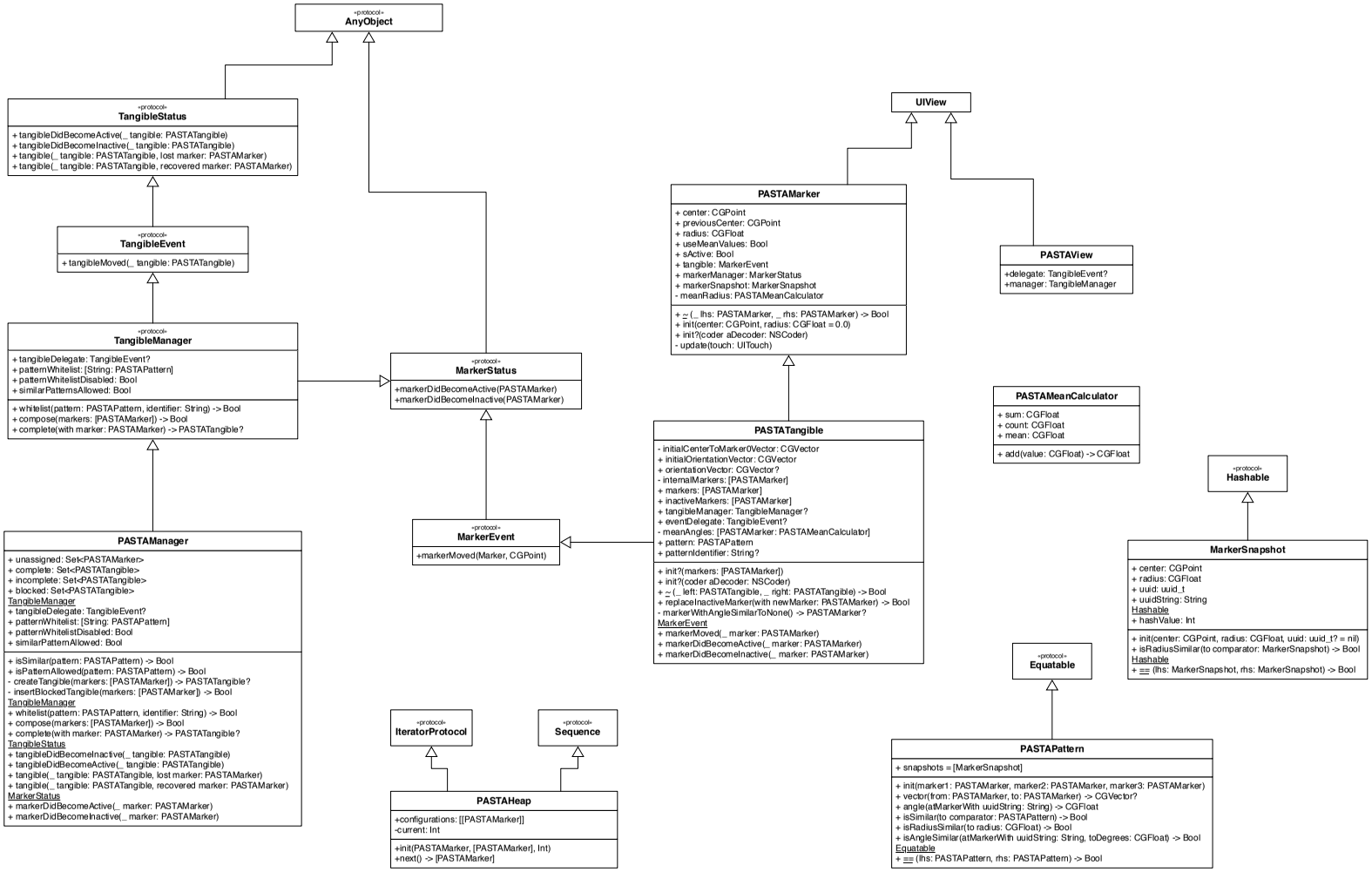

Figure 5.1: Diagram showing the inheritance of PASTA classes.

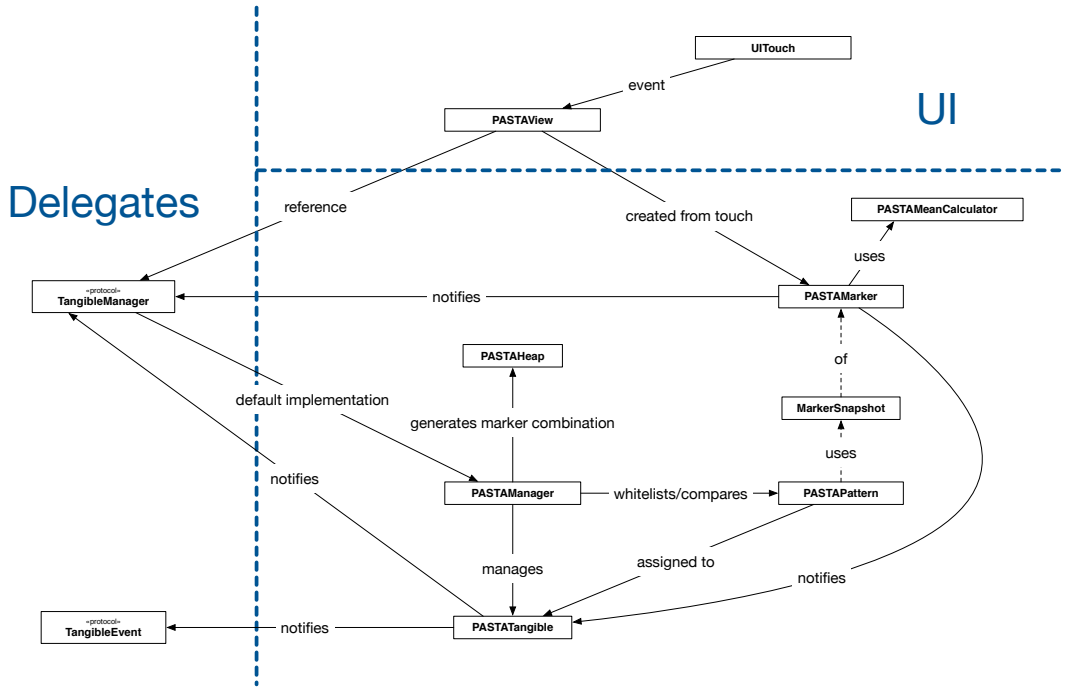

Figure 5.2: Everything starts with a `UITouch` event received by `PASTAView`. `PASTAMarker` are created from this touch and `PASTAManager` takes over to manage the marker. The manager tries to compose/complete a `PASTATangible` and manages identifying/blocking via `PASTAPattern`. The tangible itself notifies the manager and the delegate (`TangibleEvent`) of its changes.

5.2.1 `PASTAView`

This is the root of the SDK. It inherits from `UIView` and overrides `hitTest(_:with:)`. The method will either return the sub view at the hit position if one exists, otherwise a new instantiated `PASTAMarker`, but never itself. The marker is also added as a sub view of `PASTAView`. Using this approach it saves up the manual distribution of touch events to the appropriate marker because the system now delivers `UITouch` directly to the marker.

To receive updates about tangibles implement the `TangibleEvent` protocol and assign it to the `delegate` property.

By default `PASTAManager` is used as marker and tangible

manager. If you like to implement your own manager implement the `TangibleManager` protocol and assign it to the `manager` property.

5.2.2 `PASTAMarker`

This class represents the markers of a `PASTATangible`. It also inherits from `UIView` and overrides `touchesBegan(_:with:)`, `touchesEnded(_:with:)`, `touchesCancelled(_:with:)`, `touchesMoved(_:with:)` of `UIResponder` to react to touch events.

A marker is becoming active after `touchesBegan(_:with)` is called and inactive after `touchesEnded(_:with)` or `touchesCancelled(_:with)` is called. If assigned to a tangible the view will not be removed from its super view after the touch ended or was canceled. Because of that the marker can still receive a new touch and turn active without involving any additional class.

A marker has two main properties `radius` and `center`. The latter is used from superclass `UIView`. The `radius` reflects the value `majorRadius` property of `UITouch`. The `width` and `height` of the view is set to `2 * radius`. Therefore, no other touch can occur on the view since it has the same size as the touch.

By default the value of `radius` is averaged to make it more robust against outliers. It can be disabled by setting `useMeanValues` to `false`.

The class provides an implementation of the $\sim$ infix operator. The method compares the radii of two markers using their `markerSnapshot` property. See 5.2.6 "`MarkerSnapshot`" for more details.

5.2.3 `PASTATangible`

A `PASTATangible` inherits from `PASTAMarker` and implements `MarkerEvent` (5.2.9 "Protocols"). Therefore,

each tangible could be used as a marker if desired.

The `center` of the tangible is the circumcenter of the triangle formed by the three markers. The `radius` is the distance between `center` and a marker.

Orientation In addition to a marker's main properties a tangible has an `orientationVector` and a `pattern`. The `orientationVector` does what the name already states: it is a normalized vector facing into the direction the tangible is rotated. The property is an *optional* because it depends on the pattern of the tangible whether one marker can be distinguished from the others. If you want the rotation since initialization of the tangible you should use `initialOrientationVector`.

Pattern The `pattern` property (5.2.5 "`PASTAPattern`") contains the arrangement of markers. It is used to distinguish a tangible from others and helps handle inactive markers which will be explained later in Section "Moving with Inactive Markers". The `pattern` reflects the last state of the tangible where all markers were active. If the whitelist of the `PASTAManager` contains a pattern which is similar to the pattern of this tangible the property `patternIdentifier` will return the identifier of the whitelisted pattern.

All markers assigned to a tangible will notify it through the `MarkerEvent` protocol.

If a marker turned inactive the tangible is able to tackle this loss. A tangible's state is active as long as it has one active marker. How this loss is handled is explained in the following sub sections.

Moving with Inactive Markers

There are two different scenarios. We can either have one or two inactive markers.

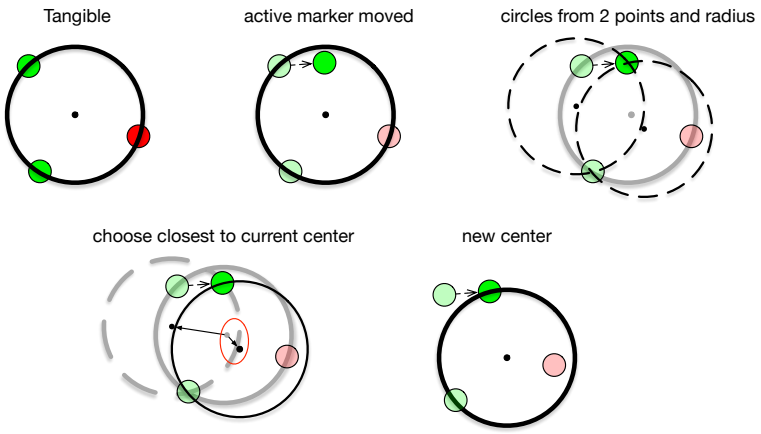

Figure 5.3: Visualization of finding new center with two active (green circles) and one inactive marker (red circle).

Two Inactive Markers This scenario is trivial. We just calculate the translation of the single active marker and apply it to the two inactive markers and the tangible's center.

One Inactive Marker This scenario is more complicated. We cannot just apply the translation of the moved marker to the inactive. If the inactive marker is translated every time an active marker moved the inactive marker would move twice as often because both active markers can move. One also has to take into account that the active markers are moving one after another and not in parallel.

Therefore we have to take the position of both active markers into account. First we calculate the centers of the two possible circles which can be created using the current `radius` and the two active markers. The appropriate function is `circleCenters(point1:point2:radius:)` which is an extension to `CGPoint` and is based on a solution of The Math Forum [2017]. We choose the center which is closest to the current center as our new center position of the tangible. See Figure 5.3 for a visualization.

Now that we have our new center we have to calculate the new position of the inactive marker. Have a look at Figure 5.4 for a visualization. For this we are using the

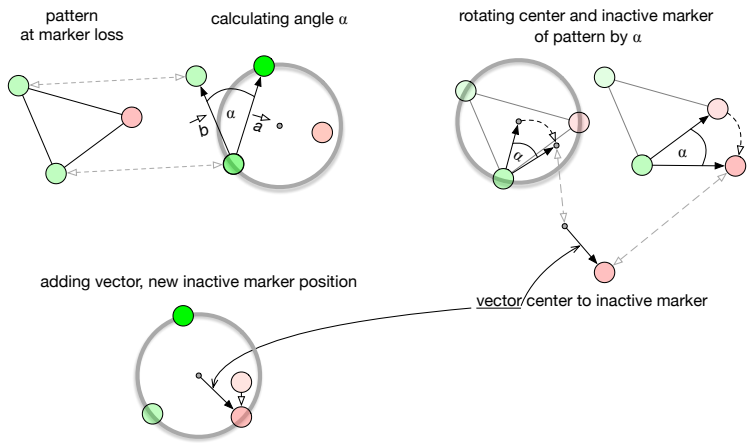

Figure 5.4: Visualization of new position calculation with only one inactive marker (red circle).

`pattern` of the tangible. First, we create two vectors. A vector $\vec{a}$ from the *other active marker* to *moved marker* and a vector $\vec{b}$ using the appropriate positions of the pattern.

$$\vec{a} = active \rightarrow moved \tag{5.1}$$

$$\vec{b} = active_{pattern} \rightarrow moved_{pattern} \tag{5.2}$$

Then we calculate the angle α between those two vectors.

$$\alpha = \vec{a} \measuredangle \vec{b} \tag{5.3}$$

To find the approximate new position we rotate the inactive marker and center **of the pattern** by angle α around the *other active marker* (Figure 5.4, top right). The vector facing from the rotated center to the rotated inactive marker in the pattern needs to be added to the current center of the tangible shown in the lower left part of Figure 5.4. This will be our new position of the inactive marker.

The only error we get is the varying distance between the two active markers which will be just some points because the pattern of tangibles are fixed. On that basis, we assume that the error is negligible.

Replacing an Inactive Marker

Again, we have to differentiate between one or two inactive markers.

Replacing Single Inactive Markers We take the current two active markers with the new marker and create a `PASTAPattern` of it. Then we compare the pattern with the pattern of the tangible using `isSimilar(to:)`. If `true` we will use the new marker as a replacement for the inactive marker.

Replacing One of Two Inactive Marker In this scenario we have just one active and two inactive markers. To achieve this we compare the distance between the active and the new marker ($\vec{a}$) with the distance between the active marker and the two inactive markers in the pattern ($\vec{i_n}$). If the $offset$ between them is smaller than or equal to the radius of the inactive marker $inactive_{n,pattern}$ we have a candidate. If both inactive markers could be replaced we choose the one which is closer to the new marker. See Figure 5.5.

First we create the vector $\vec{a}$ from the active marker to the new one.

$$\vec{a} = active \rightarrow new \tag{5.4}$$

For each inactive marker $inactive_n$ we now form a vector $\vec{i}$ from the active marker by taking the positions in the pattern.

$$\vec{i_n} = active_{pattern} \rightarrow inactive_{n,pattern} \tag{5.5}$$

We then compute the offset of the length of $\vec{a}$ and $\vec{i_n}$.

$$offset = |\frac{\vec{a}}{||\vec{a}||} - \frac{\vec{i_n}}{||\vec{i_n}||}| \tag{5.6}$$

$$\tag{5.7}$$

If $offset$ is less than or equal to $radius$ we have a candidate.

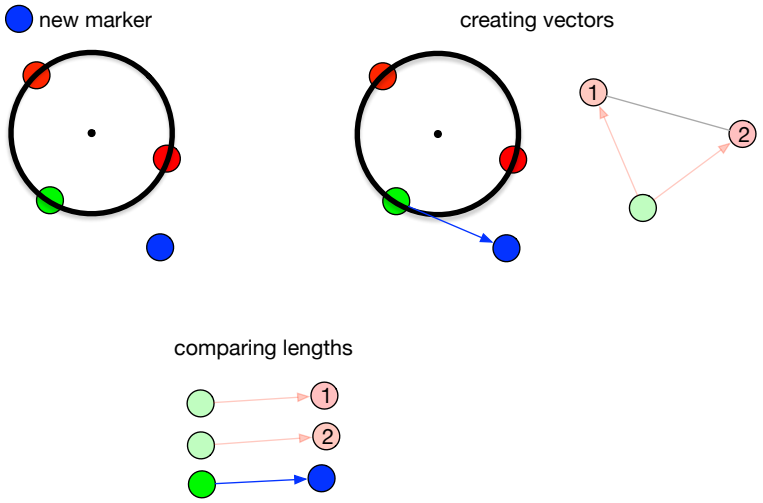

Figure 5.5: Visualization of inactive marker replacement with two inactive markers (red circles).

5.2.4 `PASTAManager`

The `PASTAManager` manages all markers of the appropriate `PASTAView` by either trying to complete or to compose tangibles. The class also manages the identification of tangibles through a set of patterns. It implements the `TangibleManager` protocol and is the default manager of every `PASTAView`.

To receive updates about tangibles like newly detected or lost ones, markers lost or recovered, and tangible position changes you have to implement the `TangibleEvent` protocol and assign the object to `tangibleDelegate`. Setting `PASTAView.delegate` does exactly the same.

Tangible Identification The property `patternWhitelist` describes a whitelist of `PASTAPatterns` which will be recognized. By default only tangibles with a pattern similar to the ones in `patternWhitelist` are detected. Use `whitelist(pattern:identifier:)` to add a pattern to the list with an identifier of your choice. If a similar pattern or an equal identifier is already contained

in the list nothing will be added. Each composed tangible which is similar to a white-listed pattern will receive the identifier by setting the `patternIdentifier` property of `PASTATangible`. If the whitelist is empty no tangible will be recognized. To disable the whitelist set the property `patternWhitelistDisabled` to `true` for example to insert patterns during runtime. Have a look at the example application included in the PASTA SDK.

Additionally, if a new tangible is detected with a pattern similar to any active on the screen it will be blocked as well. You can allow similar patterns by setting the property `similarPatternAllowed` to `true`.

Sets `PASTAManager` has several set properties to keep track of markers and tangibles. The `unassigned` property is a set of markers which have not been assigned to a tangible yet. `complete` and `incomplete` sets contain all current active tangibles on the screen which are white-listed and not similar to any other. Both sets are disjunct. `blocked` is a set of currently active tangibles which are either not allowed (`patternWhitelistDisabled = false`) or too similar to others (`similarPatternAllowed = false`). Blocked tangibles are hidden from the developer and therefore will never occur as a parameter of the `tangibleDelegate` functions. Tangibles are blocked so that their markers do not interfere with others.

Receiving a new Marker

The main logic takes part in `markerDidBecomeActive(_:)`. See Algorithm 4 for a pseudo code implementation. First we check whether a tangible can be completed with the new marker. If so, we are done. Otherwise, we check the if location of the marker is inside an `incomplete` or `blocked` tangible. If the marker is inside we know that a tangible did not accept the marker as a replacement but it definitely belongs to a tangible. In this case we just return and do not add the marker to the set of `unassigned`

markers. If we have more than two unassigned markers we can try to compose a tangible. If none tangible could be composed we add the marker to the set of `unassigned` markers. Otherwise we will remove all markers used to compose the tangible from the `unassigned` set.

Algorithm 4 `markerDidBecomeActive(_:)`

```
1: if marker completes a tangible then
2:     marker.markerManager ← nil
3:     return
4: end if
5:
6: markerInside ← marker inside incomplete/blocked tangible
7: if markerInside then
8:     return
9: end if
10:
11: if unassigned.count ≥ 2 then
12:     patterns ← PASTAHEAP(marker, unassigned)
13:     for all pattern in patterns do
14:         isComposed ← COMPOSE(pattern)
15:         if isComposed then
16:             ...
17:             return
18:         end if
19:     end for
20: end if
21:
22: insert marker into unassigned
```

Composing a Tangible

Algorithm 5 shows the pseudo code of composing a tangible. The relevant part happens in line 6 and 7. If the statement in line 6 evaluates to `true` we will create a 'blocked' tangible. The difference to a 'normal' tangible is that we add it to the `blocked` set and do not notify the `tangibleDelegate`. Especially, `compose(markers:)` will return `true` to force `markerDidBecomeActive(_:)` to remove the markers from the `unassigned` set.

Algorithm 5 `compose(markers:)`

1: **if** $markers.count \neq 3$ **then**
2: **return** $false$
3: **end if**
4:
5: $pattern \leftarrow$ PASTAPATTERN($markers$)
6: **if** $pattern$ not allowed $\vee$ (similar pattern exists $\wedge$ similar pattern not allowed) **then**
7: INSERTBLOCKEDTANGIBLE(markers)
8: **return** $true$
9: **end if**
10:
11: $tangible \leftarrow$ CREATETANGIBLE($pattern$)
12: add to $complete$ (or $incomplete$) set
13:
14: $tangibleDelegate$.TANGIBLEDIDBECOMEACTIVE($tangible$)
15: **return** $true$

5.2.5 `PASTAPattern`

This class is used to describe a pattern of a tangible. It is used to compare two tangibles as well as functions as a static pattern reference for incomplete tangibles.

The previously measured error values in Chapter 4 "Analysis and Evaluation" are used in the comparison methods to counter measurement inaccuracy.

The triangle formed by the three input markers is translated such that the circumcenter is at position $(0, 0)$ of the coordinate system. This makes further calculations and comparisons easier because then all patterns share the same center position.

Internally, snapshots (`MarkerSnapshot`) of `PASTAMarker`s are used to represent a marker. Otherwise we would have a reference to the marker itself which is continuously changing. This would influence the pattern.

Differentiating Patterns

We will now explain the method `isSimilar(to:)` in more detail. The method takes into account the radius of the pattern, the internal angles of the triangle formed by the markers, and the size of the markers[5]. Algorithm 6 shows the implementation as pseudo code. First we compare the radius by evaluating `abs(tangibleA.radius - tangibleB.radius) <= allowed error`. To compare the angles and marker sizes it needs more effort. We will explain it using the angles because marker size is compared the same way just with different values. First we have to align both triangles at one corner. This is done by just comparing the values at the same index in the array of both triangles. We do not have to align them geometrically. The algorithm compares the markers in the array from top to bottom. If angles are not similar we move the first marker to the end of the array. For an example look at Table 5.1. This is done until we find similar angles or compared all markers against each other.

Tangible A	Tangible B		
marker 1	marker 1	marker 2	marker 3
marker 2	marker 2	marker 3	marker 1
marker 3	marker 3	marker 1	marker 2

Table 5.1: Pattern marker order for angle/size comparison.

5.2.6 `MarkerSnapshot`

The `MarkerSnapshot` struct represents a marker at a specific point in time. It has properties for `center` and `radius`. On initialization a UUID (Universally Unique Identifier) is generated if none is provided. This UUID is used to later reference a marker snapshot with a `PASTAMarker` in the pattern.

[5] In the current implementation the size comparison of the markers is disabled because they provide discrete and not continuous values (4.2 "Marker Size".)

Algorithm 6 `isSimilar(to:)`

```
1: isSimilar ← false
2: if radiusA not similar to radiusB then
3:     return false
4: end if
5: isSimilar ← true
6: for all markers in a tangible do
7:     for all markersA, markersB do
8:         if markerA angle not similar to markerB then
9:             isSimilar ← false
10:            break
11:        end if
12:        if markerA radii not similar to markerB then
13:            isSimilar ← false
14:            break
15:        end if
16:    end for
17:    if isSimilar then
18:        break
19:    end if
20:    move first markerB to end of array
21: end for
22: return isSimilar
```

The struct also provides the method `isRadius Similar(to:)` to compare the radii of two snapshots.

5.2.7 `PASTAHeap`

This class conforms to the `Sequence` and `Iterator-Protocol`. Therefore instances of `PASTAHeap` can be used with the `[]` subscript notation.

Initializing using `init(marker:unassigned: markerPerPattern:)` generates an array containing arrays of `PASTAMarker`. Each marker array has a default size of three (`markerPerPattern`) and contains the marker of the `marker` parameter. The remaining two fields of a marker array are filled with a combination of two markers provided by the `unassigned` parameter.

This results in a total of

$$\frac{n!}{k!(n-k)!} = \binom{n}{k} \tag{5.8}$$

combinations where n is the number of `unassigned` markers and $k = \texttt{markerPerPattern} - 1$.

5.2.8 `PASTAMeanCalculator`

It calculates the mean of all values added to an instance of `PASTAMeanCalculator`. It is used in `PASTAMarker` and since `PASTATangible` is a subclass in that class as well.

Within the same file there is a second class; `Counter`. It counts the frequencies of each value which was added to it and returns the most frequent value. The class is currently unused. The idea is to replace the mean calculator of `PASTAMarker` with the `Counter` class because radius values of `UITouch` are discrete.

5.2.9 Protocols

We will now mention each protocol and the purpose. Check the documentation in the code for a more detailed description.

`MarkerStatus` The protocol provides two methods informing about the state of a marker. Active stands for a marker which `UITouch` began. Inactive is a marker which `UITouch` ended or was canceled.

`MarkerEvent` The protocol inherits from `MarkerStatus`. It has a method which notifies that a marker moved.

`TangibleStatus` This protocol notifies about tangible status changes. Additionally, it forwards marker status changes by telling that a tangible recovered or lost a marker.

`TangibleEvent` Inherits from `TangibleStatus` and adds an additional method notifying about marker movements.

`TangibleManager` Describes common properties and methods every tangible manager should have. Implement this protocol to exchange the existing manager with your own implementation.

5.3 Limitations

It is currently not possible to use the marker size as a parameter to differentiate tangibles. This limitation is based on the findings in Section 4.2 “Marker Size”. Suggested improvements are described in Section 6.2.1 “Improving PASTA SDK”.

Chapter 6

Summary and Future Work

6.1 Summary and Contributions

We investigated parameters of passive tangibles on capacitive screens capable of uniquely identifying a tangible and to determine its rotation. We tested tangibles with three markers using different diameters, interior angles, and marker sizes. Results showed that markers with sizes greater than or equal to 15 mm should not be used because results show that they sometimes generate two touch events.

Do not use marker sizes ≥ 15 mm.

The error of the tangible radius that should be taken into account when building or distinguishing tangibles is 1.3 millimeters. To identify the rotation we used the interior angles that must not be symmetric. The error taken into account to differentiate them is 12.396 degrees.

Error values of tangible radius and interior angles.

We developed an iOS SDK called PASTA that is capable of identifying tangibles and detecting their rotation using the aforementioned error values. Marker size is currently not used to differentiate or detect rotation since `UITouch.majorRadius` values are discrete and not continuous. Implementing a detector which tackles the marker

The PASTA SDK; capable of identifying and detecting rotation of passive tangibles.

size detection problem would exceed this thesis and is left to future work. The problem and possible approaches to solve it are described in more detail in Section 6.2.1 "Improving PASTA SDK".

6.2 Future Work

New conductive materials should be reviewed such as conductive foam [we online.de, 2017a] or Conductive Graphene Filament [Engineering.com, 2017] to investigate whether they provide more reliable results.

Evaluate detection rate of the SDK.

In a next step, the detection rate of the PASTA SDK needs to be statistically evaluated. Type 1 (false-positive) und type 2 (false-negative) errors should be taken into account. We did some minor qualitative tests during development by placing tangibles on the screen to check the expected behavior.

Conduct a user study with developers.

PASTA is developed to make it easy for developers to create iOS applications using passive tangibles. To validate the SDK a user study should be conducted with developers containing a programming task.

Trying to minimize interior angle error.

The analyzed interior angle error of 12.396 degrees is quite high. We suggest to use the mean to minimize the error since it represents the angle[1] quite well. If so, we could use the standard error calculated by the confidence interval instead of the standard deviation that we assume to be smaller. The PASTA SDK needs to be altered to account for using the mean.
A problem arises following this approach. Currently, tangibles are immediately detected if three touches are found. To identify a tangible it needs to be compared to saved patterns which is also done immediately after the tangible is formed. This creates the problem that only one value for each angle is calculated at that time and the standard error may not reflect the right tolerance.

[1] The idea could be applicable to other parameters as well.

6.2.1 Improving PASTA SDK

Using passive tangibles, it is not possible to determine whether a user lifted the tangible or it was filtered out by the tracking technology. To approach this problem a so called *zombie mode* can be implemented. If zombie mode is enabled it keeps the representation of tangibles and markers on the screen after all markers vanished. It can be reactivated with a tangible having the right pattern if placed with one of his markers at least on one zombie marker representation. To remove the zombie tangible one can use a double tap or a swipe gesture on the tangible representation itself.

Tackle the problem of filtered out tangibles with a *zombie mode*.

At the moment, the SDK is only available via CocoaPods. CocoaPods itself recommends that libraries should also support the Swift Package Manager.

Support Swift Package Manager.

We use the circumcenter computed from the positions of the three markers as the center of a tangible. As size we use the distance between the center and one of the markers (radius). We did not take into account tangibles which have the size of a triangle formed by the three markers. The centroid or incenter of the triangle can also be used as the center of a triangle. Then, a new value for the size of the tangible has to be found since the distance to each point from the center is not equal.

Add support of non-round tangibles.

Linden [2015] implemented a deformation check to detect tangibles which markers are moving away from each other. This can happen if three fingers are detected as a tangible. It can also happen if two tangibles are placed on the screen almost simultaneously. Two markers of the first tangible and then one marker of the later tangible may be recognized before all three markers of the first tangible are recognized. The first three will then form a tangible because every tangible initialized with three markers is created. Currently, no checks are implemented like a too small or too big diameter.

Add a deformation check whose checks for markers moving away from each other.

Adaption of the TUIO protocol would allow the iPad to be used by other systems and applications as a tangible user interface.

Adapt the TUIO protocol.

Solving the Marker Size Problem As mentioned, the touch radius values provided by iOS are discrete and not continuous. This implies, that they are jumping by fixed values which makes it very unreliable. We showed that the measured radii for marker sizes greater than or equal to 8 mm are not detected as one value. Instead, they were detected as two or three different radii. There are currently two unsolved problems:

- When detection begins, markers may receive a radius of 10.42 points which is the lowest possible value. However, their actual size might be bigger and their measured radius will change after some milliseconds. Immediately forming a tangible and comparing it with another one will lead to false results.
- The other problem occurs later when, for example, two tangibles are compared. It may happen that a marker's radius is detected with a lower or higher value at the time it is compared to another one which again leads to a false result.

Count radius values over time and use the most frequent one.

For the latter problem we recommend a counter that sets (or returns) the radius value of a marker to the most frequent value occurring over time. It implies that all measured values have to be counted throughout the lifetime of a marker. The first problem is not a trivial problem. A solution could be letting the marker gather some data points before it notifies the marker manager of its presence. But this implies a delay.

Bibliography

Apple. Apple developer documentation, 2017a. URL https://developer.apple.com/documentation. Retrieved November 27, 2017.

Apple. Apple pencil, 2017b. URL https://www.apple.com/apple-pencil/. Retrieved November 27, 2017.

Apple. Identify your ipad model, 2017c. URL https://support.apple.com/en-us/HT201471. Retrieved November 27, 2017.

Apple. Apple ipad pro, 2017d. URL https://www.apple.com/eg/ipad-pro/specs/. Retrieved November 27, 2017.

Andrea Bottino, Andrea Martina, Francesco Strada, and Amirhosein Toosi. GAINE - A portable framework for the development of edutainment applications based on multitouch and tangible interaction. *Entertainment Computing*, 16:53–65, jul 2016. ISSN 18759521. doi: 10.1016/j.entcom.2016.04.001. URL http://linkinghub.elsevier.com/retrieve/pii/S187595211630012X.

Liwei Chan, Stefanie Müller, Anne Roudaut, and Patrick Baudisch. CapStones and ZebraWidgets: sensing stacks of building blocks, dials and sliders on capacitive touch screens. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*, (c):2189, 2012. doi: 10.1145/2207676.2208371. URL http://dl.acm.org/citation.cfm?doid=2207676.2208371http://dl.acm.org/citation.cfm?id=2207676.2208371.

Raffaele Di Fuccio, Giovanni Siano, and Antonio De Marco. TriPOD: A Prototypal System for the Recognition of Capacitive Widget on Touchscreen Addressed for Montessori-Like Educational Applications. pages 664–676. Springer, Cham, apr 2017. doi: 10.1007/978-3-319-56538-5_68. URL http://link.springer.com/10.1007/978-3-319-56538-5{_}68.

Brien East, Sean DeLong, Roozbeh Manshaei, Ahmed Arif, and Ali Mazalek. Actibles: Open Source Active Tangibles. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces - ISS '16*, pages 469–472, New York, New York, USA, 2016. ACM Press. ISBN 9781450342483. doi: 10.1145/2992154.2996874. URL http://dl.acm.org/citation.cfm?doid=2992154.2996874.

Eclipse. Jetty, 2017. URL https://www.eclipse.org/jetty/. Retrieved November 27, 2017.

Eisenmax. Blei feinschrot ø 0,6 - 1,5 mm, 2017. URL http://www.eisenmax.com/eisen-p385h130s142-010-Blei-Feinschrot-.html?sid=f2dcac11f72c0deec1a53d93d27168f6. Retrieved November 27, 2017.

Engineering.com. Conductive graphene filament for 3d printing, 2017. URL https://www.engineering.com/3DPrinting/3DPrintingArticles/ArticleID/9797. Retrieved November 27, 2017.

Andy P. Field. *Discovering statistics using SPSS : (and sex and drugs and rock 'n' roll)*. SAGE Publications, 2009. ISBN 1847879071. URL https://books.google.de/books/about/Discovering{_}Statistics{_}Using{_}SPSS.html?hl=de{&}id=a6FLF1YOqtsC.

G.W. Fitzmaurice, Hiroshi Ishii, and W.a.S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 442–449, New York, New York, USA, 1995. ACM Press. ISBN 0201847051. doi: 10.1145/223904.223964. URL http://portal.acm.org/citation.cfm?doid=223904.223964.

GestureWorks. Gesture and multitouch software authoring sdk, 2017. URL http://gestureworks.com/. Retrieved October 24, 2017.

Asghar Ghasemi and Saleh Zahediasl. Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, 10(2): 486–489, 2012. ISSN 1726913X. doi: 10.5812/ijem.3505. URL http://www.ncbi.nlm.nih.gov/pubmed/23843808http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3693611.

Timo Götzelmann and Daniel Schneider. CapCodes: Capacitive 3D Printable Identification and On-screen Tracking for Tangible Interaction. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction - NordiCHI '16*, pages 1–4, New York, New York, USA, 2016. ACM Press. ISBN 9781450347631. doi: 10.1145/2971485.2971518. URL http://dl.acm.org/citation.cfm?doid=2971485.2971518.

Toine Heuvelmans. Geometry, simplified, 2017. URL https://github.com/toineheuvelmans/Metron. Retrieved November 27, 2017.

Hiroshi Ishii. The tangible user interface and its evolution. *Communications of the ACM*, 51(6):32–36, jun 2008. ISSN 00010782. doi: 10.1145/1349026.1349034. URL http://portal.acm.org/citation.cfm?doid=1349026.1349034.

Hiroshi Ishii and Brygg Ullmer. Tangible bits. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, pages 234–241, 1997. doi: 10.1145/258549.258715. URL http://portal.acm.org/citation.cfm?doid=258549.258715.

Martin Kaltenbrunner. reacTIVision and TUIO: a tangible tabletop toolkit. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces - ITS '09*, page 9, 2009. doi: 10.1145/1731903.1731906. URL http://portal.acm.org/citation.cfm?doid=1731903.1731906.

Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. TUIO: A protocol for table-top tangible

user interfaces. *6th International Gesture Workshop*, pages 1–5, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.6437http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.155.3890{&}rep=rep1{&}type=pdf.

Martin Kaltenbrunner, Sergi Jorda, Gunter Geiger, and Marcos Alonso. The reacTable*: A Collaborative Musical Instrument. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*, pages 406–411. IEEE, 2006. ISBN 0-7695-2623-3. doi: 10.1109/WETICE.2006.68. URL http://ieeexplore.ieee.org/document/4092244/.

Sven Kratz, Tilo Westermann, Michael Rohs, and Georg Essl. CapWidgets: tangile widgets versus multi-touch controls on mobile devices. *Proceedings of CHI 2011*, pages 1351–1356, 2011. doi: 10.1145/1979742.1979773. URL http://portal.acm.org/citation.cfm?doid=1979742.1979773http://dl.acm.org/citation.cfm?id=1979773{%}5Cnpapers://c80d98e4-9a96-4487-8d06-8e1acc780d86/Paper/p17197.

LEGO. Lego mindstorms ev3, 2017a. URL lego.com/en-us/mindstorms/products/mindstorms-ev3-31313. Retrieved November 27, 2017.

LEGO. Lego mindstorms user guide, 2017b. URL https://www.lego.com/de-de/mindstorms/downloads/user-guide/. Retrieved November 27, 2017.

leJOS. Java for lego mindstorms, 2017. URL http://www.lejos.org/ev3.php. Retrieved November 27, 2017.

Rong-Hao Liang, Kai-Yin Cheng, Liwei Chan, Chuan-Xhyuan Peng, Mike Y. Chen, Rung-Huei Liang, De-Nian Yang, and Bing-Yu Chen. GaussBits: Magnetic Tangible Bits for Portable and Occlusion-Free Near-Surface Interactions. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, pages 1391–1400, 2013. doi: 10.1145/2470654.2466185. URL http://dl.acm.org/citation.cfm?doid=2470654.2466185.

Rong-Hao Liang, Liwei Chan, and Hung-yu Tseng Hanchih Kuo. GaussBricks: Magnetic Building Blocks for Constructive Tangible Interactions on Portable Displays. *Proc. CHI'14*, (Figure 1):3153–3162, 2014. ISSN 9781450324748. doi: 10.1145/2556288.2557105. URL http://dl.acm.org/citation.cfm?doid=2556288.2557105.

René Linden. *Multitouchkit: A Software Framework for Touch Input and Tangibles on Tabletops and Mobile Devices*. PhD thesis, RWTH Aachen University, 2015. URL http://hci.rwth-aachen.de/materials/publications/linden2015a.pdf.

LRI. Laboratoire de recherche en informatique - touchtokens, 2017. URL https://www.lri.fr/~appert/touchtokens/. Retrieved November 27, 2017.

mathforum.org. The math forum - finding the center of a circle from 2 points and radius, 2017. URL http://mathforum.org/library/drmath/view/53027.html. Retrieved November 27, 2017.

mathopenref.com. Math open reference - circumcircle of a triangle, 2017. URL https://www.mathopenref.com/trianglecircumcircle.html. Retrieved November 27, 2017.

Microsoft. Microsoft surface dial, 2017a. URL https://www.microsoft.com/en-us/surface/accessories/surface-dial. Retrieved November 27, 2017.

Microsoft. Microsoft surface hub, 2017b. URL https://www.microsoft.com/microsoft-surface-hub. Retrieved November 27, 2017.

Rafael Morales González, Caroline Appert, Gilles Bailly, and Emmanuel Pietriga. TouchTokens: Guiding Touch Patterns with Passive Tokens. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 4189–4202, 2016. doi: 10.1145/2858036.2858041. URL http://dl.acm.org/citation.cfm?doid=2858036.2858041.

Claire O'Malley, S. Fraser, and Others. Literature review in learning with tangible technologies. *Halshsarchivesouvertesfr*, pages 1–52, 2004. ISSN 9780954859428. doi: papers2://publication/uuid/EDD99909-5D5D-4177-8916-B3C1803CFF8B. URL https://hal.archives-ouvertes.fr/hal-00190328/ http://hal.archives-ouvertes.fr/hal-00190328/{%}5Cnhttp://halshs.archives-ouvertes.fr/hal-00190328/.

onmsft.com. 5 cool facts about microsoft's new surface dial, 2017. URL https://www.onmsft.com/news/5-cool-facts-microsoft-surface-dial. Retrieved November 28, 2017.

opencsv. Easy-to-use csv (comma-separated values) parser library for java, 2017. URL http://opencsv.sourceforge.net/. Retrieved November 27, 2017.

Oracle. Json-p - java api for json processing, 2017a. URL https://javaee.github.io/jsonp/. Retrieved November 27, 2017.

Oracle. Remote method invocation, 2017b. URL http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html. Retrieved November 27, 2017.

pandas. Open source, bsd-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the python programming language, 2017. URL http://pandas.pydata.org. Retrieved November 27, 2017.

Ben Piper, Carlo Ratti, and Hiroshi Ishii. Illuminating Clay : A 3-D Tangible Interface for Landscape Analysis. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '02*, (1):355–362, 2002. ISSN 10878270. doi: 10.1145/503376.503439. URL http://portal.acm.org/citation.cfm?doid=503376.503439.

Programiz. Python program to calculate the area of a triangle, 2017. URL https://www.programiz.com/python-programming/examples/area-triangle. Retrieved November 27, 2017.

Patrick Royston. Approximating the Shapiro-Wilk W-test for non-normality. *Statistics and Computing*, 2(3): 117–119, sep 1992. ISSN 09603174. doi: 10.1007/BF01891203. URL http://link.springer.com/10.1007/BF01891203.

SAS. Statisticaldiscovery. from sas., 2017. URL https://www.jmp.com/. Retrieved November 27, 2017.

Bertrand Schneider, Patrick Jermann, Guillaume Zufferey, and Pierre Dillenbourg. Benefits of a tangible interface for collaborative learning and interaction. *IEEE Transactions on Learning Technologies*, 4(3):222–232, jul 2011. ISSN 19391382. doi: 10.1109/TLT.2010.36. URL http://ieeexplore.ieee.org/document/5654494/.

Statista.com. The Statistics Portal for Market Data, Market Research and Market Studies, 2017. URL https://www.statista.com/. Retrieved October 26, 2017.

techradar. The tesla s has the most insane in-car touchscreen multimedia system ever, 2017. URL http://www.techradar.com/news/car-tech/the-new-tesla-s-has-the-most-insane-in-car-touchscreen-multimedia-system-ever-1185159. Retrieved November 27, 2017.

Philip Tuddenham, David Kirk, and Shahram Izadi. Graspables Revisited: Multi-Touch vs. Tangible Input for Tabletop Displays in Acquisition and Manipulation Tasks. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, page 2223, 2010. doi: 10.1145/1753326.1753662. URL http://portal.acm.org/citation.cfm?doid=1753326.1753662.

Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergård, and Jan Borchers. Pucs: Detecting transparent, passive untouched capacitive widgets on unmodified multi-touch displays. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 101–104, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2271-3. doi: 10.1145/2512349.2512791. URL http://doi.acm.org/10.1145/2512349.2512791.

Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergård, and Jan Borchers. Percs: Persistently trackable tangibles on capacitive multi-touch displays. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 351–356, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.2807466. URL http://doi.acm.org/10.1145/2807442.2807466.

we online.de. Würth elektronik group - we-ls conductive foam, 2017a. URL http://katalog.we-online.de/en/pbs/WE-LS. Retrieved November 27, 2017.

we online.de. Würth elektronik group - we-lt conductive shielding gasket, 2017b. URL http://katalog.we-online.de/en/pbs/WE-LT. Retrieved November 27, 2017.

Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan, and Jan Borchers. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, page 481, New York, New York, USA, 2009. ACM Press. ISBN 9781605582467. doi: 10.1145/1518701.1518779. URL http://dl.acm.org/citation.cfm?doid=1518701.1518779http://portal.acm.org/citation.cfm?doid=1518701.1518779.

YouTube. Work smarter in the field with tablets powered by intel — intel business, 2017a. URL https://www.youtube.com/watch?v=6AJ6BwMjmX4. Retrieved November 27, 2017.

YouTube. Rolls-royce future shore control centre, 2017b. URL https://www.youtube.com/watch?v=vg0A9Ve7SxE. Retrieved November 27, 2017.

Index

Typeset November 28, 2017