FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Verwendung von taktilen Karten für Blinde

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Software Engineering & Internet Computing**

eingereicht von

**Roland Prinz**
Matrikelnummer 0726957

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, 20. April 2016

| Roland Prinz | Peter Purgathofer |
|---|---|

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Using tactile maps for the blind

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering & Internet Computing**

by

**Roland Prinz**
Registration Number 0726957

to the Faculty of Informatics

at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Vienna, 20th April, 2016

Roland Prinz

Peter Purgathofer

# Erklärung zur Verfassung der Arbeit

Roland Prinz
Staudgasse 43/10, 1180 Wien

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

Wien, 20. April 2016

Roland Prinz

# Kurzfassung

Das Ziel dieser Diplomarbeit ist es, durch die Verknüpfung von Technik und Information, blinde Personen bei der Erkundung ihrer Umgebung zu unterstützen. Der entwickelte Prototyp bietet die Möglichkeit, Stadtkarten mittels audiounterstützter Software zu erforschen. Durch die Kombination von taktilen Karten mit einem Multi-Touch Sensor, können hinterlegte kartografische Informationen durch den Einsatz von Gesten abgerufen und verarbeitet werden. Vorhandenes Umgebungswissen der AnwenderInnen wird dabei um neues, kartografisches Wissen aus dem Prototypen ergänzt. So wird das Erlernen von räumlichem Denken aktiv unterstützt.

Die Diplomarbeit besteht aus einer Analyse und Kombination von bestehenden Forschungsergebnissen und Projekte sowie der Evaluierung von Multi-Touch Technologien für den Einsatz des Prototyps. Dabei werden die Technologien Leap Motion Sensor und das Apple iPad Pro in unterschiedlichen Situationen und Anwendungsfällen für den möglichen Einsatz getestet. Im Zuge der Entwicklung wird ein Konzept erarbeitet und ein Prototyp erstellt, der die genannten Anforderungen vollumfänglich erfüllt. Bei der Entwicklung werden die Software, als auch die Erstellung der taktilen Karten und einem Gehäuse für den Prototyp, berücksichtigt. Abschließend wird im Rahmen eines Workshops das Konzept und der Prototyp durch blinde ExpertenInnen für den Einsatz in Schulklassen evaluiert.

Das Resultat sind zwei haptische Stadtkarten mit hinterlegten Informationen in den abgebildeten Bereichen und ein vollständiger Prototyp basierend auf dem Apple iPad Pro. Das Konzept erläutert alle Funktionen der Software und die Beschreibung der einzelnen Gesten zur Interaktion mit der Software. Mit diesem Konzept ist der Grundstein für ein selbstständig verwendbares Werkzeug für Personen mit einer Sehbeeinträchtigung zum Lesen von Landkarten gelegt.

# Abstract

The main goal of this master's thesis is the combination of technology and information to support blind people discovering their domestic environment. The prototype offers the possibility to explore city maps with an audio enhanced software. Due to the usage of tactile maps and a multi touch device, geobased information can be accessed using hand gestures. Through the mix of the user's knowledge of the environment and the new information gained from the prototype, learning of map reading skills and visual thinking can be supported.

This thesis consists of an analysis and consolidation from previous research findings and projects as well as the evaluation of multi-touch technologies as a basis for the prototype. Therefore the Leap Motion Sensor as well as the Apple iPad Pro will be tested in various situations and use cases for the possible applications. During the development phase a concept and a prototype will be created which fulfils the predefined requirements. The software, as well as the hardware components like the tactile maps and an iPad Pro casing will be considered. Final the thesis ends with an evaluation of the prototype with a workshop where blind experts test the usage of the device in a school environment.

The results are two tactile city maps with the needed information stored in a database as well as a fully functional prototype based on the Apple iPad Pro. The concept contains all the functions of the software and the description of the gestural interaction with the software. This concept is the basis for a self-dependent tool for reading city maps as a blind person.

# Inhaltsverzeichnis

KAPITEL 1

# Einführung

## 1.1 Einführung in das Thema

Das Thema dieser Diplomarbeit ist die Validierung von Multi-Touch Sensoren und die Implementierung eines Prototypen für das Benutzen von taktilen Karten für sehbehinderte Personen. Die Diplomarbeit basiert auf mehreren Projekten der TU Wien, die in den Jahren 2008 bis 2014 unter den Namen Sparkling Fingers[Buna] und CEAT[Arm13] in Kooperation mit dem Bundes-Blindenerziehungsinstitut durchgeführt wurden. Ziel dieser Projekte ist es, blinden Kindern räumliches Denken zu vermitteln.

Die Idee dieser Projekte ist die Entwicklung eines robusten portablen Sensors (Tablet), welcher in Interaktion mit fühlbaren Karten (Schwellkarten) geobasierte Audioinformationen ausgibt.

Zur Realisierung werden folgende Komponenten benötigt:

- **Fühlbare Karten:** Auf diesen Schwellkarten (im Folgendem auch haptische oder taktile Karte genannt) befinden sich beliebige Kartenausschnitte aus Quellen wie zum Beispiel Open Street Maps, die mittels einer eigenen Software für den Druck auf einem Braille Drucker aufbereitet werden.
- **Braille Drucker:** Mit Hilfe dieses Gerätes werden die Schwellkarten, je nach Art des Druckers, gedruckt oder gestanzt.
- **Multi-Touch Sensor:** Für die Umsetzung ist ein Multi-Touch Sensor notwendig, der Finger- und Gesteninteraktionen der benutzenden Person mit der Schwellkarte erkennt. Dabei gibt es zwei mögliche Varianten, die für diesen Zweck herangezogen werden können. Bei der Erkennung der Interaktion mittels kapazitiver Technologien wird die fühlbare Karte auf den Sensor gelegt und befestigt. Eine wesentliche Besonderheit dieser Technologie liegt dabei in der Erkennung der Eingabe, trotz der

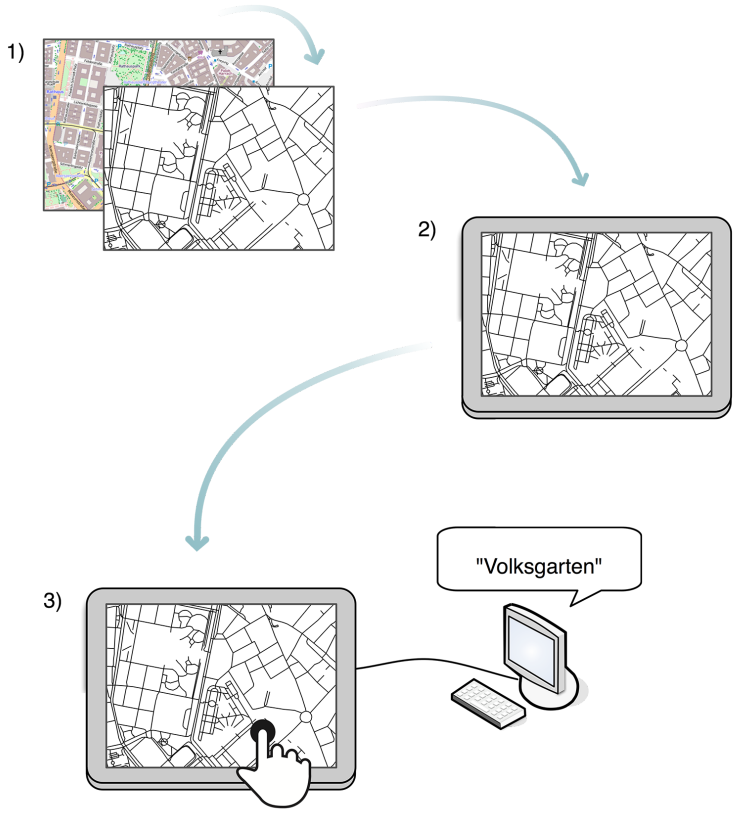


Abbildung 1.1: Eine Gesamtübersicht über das Projekt. 1) Die fühlbaren Karten werden aus Open Street Maps generiert und für den Druck auf dem Braille Drucker optimiert 2) Die Karten werden auf den Multi-Touch Sensor gelegt 3) Der/die UserIn interagiert mit den fühlbaren Karten und erhält geobasierte Audioinformationen ausgegeben.

dazwischenliegenden Karte (es besteht kein direkter Kontakt zwischen den Fingern und dem Sensor). Bei der Verwendung von Sensoren, die Finger mittels infraroten Licht und Kameras erkennen, müssen diese so angebracht sein, dass ein direkter Sichtkontakt zu den Händen und Fingern besteht.

- **Identifikation:** Um zu erkennen welche Karte auf dem Sensor liegt, wird jede Karte mit einem RFID-Chip oder einem eindeutigen Code (QR-, Strich- oder Farbcode) versehen. Diese ID wird beim Auflegen der Karte am Multi-Touch Sensor ausgelesen und interpretiert. Die Art der Identifikation richtet sich nach dem gewählten Multi-Touch Sensor und dessen Möglichkeiten.

- **Treiber & Betriebslogik des Sensors:** Der Treiber ist zuständig für die Kommunikation zwischen dem Sensor und dem Computer. Er empfängt die gemessenen

Signale und bereitet diese für die Weiterverarbeitung auf. Die Betriebslogik versucht dabei mögliche Interaktionen zu erkennen. Dabei werden Finger und Gesten erkannt, welche in Events umgewandelt und an die darauf aufbauende GeoAudioApplikation weitergeleitet werden.

- **Computer mit Audioausgabemöglichkeit:** Der Sensor ist mit einem Computer verbunden und sowohl der Treiber, die Betriebslogik als auch die GeoAudioApplikation sind darauf installiert. In der Endausbaustufe soll versucht werden den Treiber und die Betriebslogik auf den Multi-Touch Sensor laufen zu lassen, um es portabler und einfacher zu gestalten.

- **GeoAudioApplikation:** Diese Software ist dafür zuständig die Events der Betriebslogik zu empfangen und für den übermittelten Punkt auf der Karte geobasierte Audioinformationen aus verschiedensten Quellen zu suchen und auszugeben.

- **Datenplugins:** Diese Plugins werden für die Suche von geobasierten Daten verwendet. Dabei werden im Internet verschiedene Informationsquellen mittels eines eigenen Plugins ausgelesen, um die Informationen an die GeoAudioApplikation zurückzuschicken.

Eine grafische Übersicht über den möglichen Aufbau des Projektes wird in der Abbildung 1.1 gezeigt.

## 1.2 Problemstellung

Im Rahmen dieser Diplomarbeit sollen unterschiedliche Multi-Touch Sensoren getestet und für den Praxiseinsatz evaluiert werden. Im Anschluss soll eine Technologie ausgewählt und auf dieser ein Prototyp, basierend auf den Konzepten von Sommersguter[Som13] und Moser[Mos12], implementiert werden. Der finale Prototyp soll auf Praxistauglichkeit getestet und evaluiert werden.

## 1.3 Ziel der Arbeit

Ziel dieser Arbeit ist es, unterschiedliche Multi-Touch Technologien zu testen, um die beste Technologie für einen Praxistest zu identifizieren. Die dafür am besten geeignete Technologie soll im Anschluss die Basis für einen vollständigen Prototyp bilden. Das Ziel ist eine lauffähige Software zu entwickeln, welche die gewünschte Funktionalität von der Erkennung der Touches und Finger, bis hin zur Ausgabe der Audio-Informationen abdeckt. Der Einstiegspunkt in die Entwicklung ist abhängig von der gewählten Technologie und deren Schnittstellen zu wählen.

## 1.4 Methodisches Vorgehen

Die Problemstellung gliedert sich in drei Abschnitte. Der erste Teil widmet sich der Evaluierung von Multi-Touch Sensoren. Dabei werden unterschiedliche Technologien, aus dem in Kapitel 3 “Projektbezogene Technologien“ vorgestellten, ausgewählt. Für die Evaluierung dieser werden Anforderungen, basierend auf den oben genannten Konzepten, herausgearbeitet und die Multi-Touch Sensoren dagegen geprüft. Im Anschluss erfolgt die Gegenüberstellung und die Auswahl der Technologie für die Implementierung. Im zweiten Teil wird aus den Konzepten vorangegangener Diplomarbeiten ein gemeinsames Konzept für die gewählte Technologie entwickelt und diese im Anschluss implementiert. Dabei wird neben der Literaturrecherche auch auf bekannte Entwurfsmuster[1] der Software Entwicklung zurückgegriffen. Im dritten und letzten Teil wird der zuvor entwickelte Prototyp für die Verwendung in einem realen Umfeld getestet. Dafür wird ein Workshop mit blinden Personen abgehalten, die mit Aufgaben konfrontiert werden, welche es mit Hilfe des Prototypen zu lösen gilt. Anhand von dieser Methode kann das erstellte Konzept von blinden Personen, für die es erstellt wurde, getestet werden.

Alle Teile der Diplomarbeiten werden parallel durch Abstimmungs- und Designmeetings durch den Diplomarbeitsbetreuer begleitet. Die Erkenntnisse werden laufend bei den jeweiligen Phasen eingearbeitet.

## 1.5 Abgrenzung

Die Entwicklung der Hardware ist nicht Teil der Diplomarbeit. Für die Implementierung der Software wird ein fertiger und funktionsfähiger Multi-Touch Sensor verwendet. Es wird dabei auf Produkte zurückgegriffen, die sich zum aktuellen Zeitpunkt der Erstellung der Diplomarbeit am Markt befinden.

Des Weiteren ist die Erstellung der fühlbaren Karten nicht Bestandteil der Diplomarbeit. Diese werden durch das Institut, oder dessen Partner zur Verfügung gestellt. Die Modifikation zur Identifizierung dieser, kann im Rahmen der Arbeit durchgeführt werden.

Das Konzept der Implementierung wird auf den Ideen und Konzepten der Diplomarbeiten von Moser und Sommersguter aufgebaut. Es ist nicht Bestandteil der Diplomarbeit ein komplett neuartiges Konzept zu entwickeln. Verbesserungen und Erweiterungen auf Grund von Erfahrungen bei der Entwicklung oder der Praxistests werden berücksichtigt.

## 1.6 Motivation

Dieses Diplomarbeitsthema birgt sehr viel Motivation in sich. Es wird versucht ein durchaus bekanntes Problem mit Hilfe der Informatik auszugleichen und blinden Personen die Chance gegeben, ihre Umgebung, Stadt und Land näher kennen zu lernen und auf eine neue Weise zu erkunden. Mit Hilfe dieses Projektes können blinde Personen die

[1] in der Englischen Sprache auch als “Design Patterns“ bekannt

schwierigen Herausforderung ihrer Blindheit, Informationen über ihre Umgebung zu sammeln, ein bisschen ausgleichen. Sie können Auskünfte über Geschäfte, Öffnungszeiten, behindertengerechte Wege und Transportmöglichkeiten suchen und dadurch ihr räumliches Denkvermögen schulen.

Für den Multi-Touch Sensor wird auf bestehende Hardware zurückgegriffen, die keine aufwendige Modifikation benötigt.

Die dafür notwendigen Schwellkarten können beliebig erstellt werden. Es muss lediglich der gewünschte Bereich der Karte auf Open Street Maps ausgewählt werden. Dadurch kann individuell auf die Bedürfnisse der einzelnen Menschen eingegangen und für sie relevante Informationen geliefert werden. Ein weiterer Vorteil dieses Ansatzes ist die weltweite Einsatzmöglichkeit, sowie der Austausch der Karten untereinander.

## 1.7 Gliederung der Arbeit

In Kapitel 2 werden relevante Arbeiten, die bereits zu diesen oder ähnlichen Projekten durchgeführt wurden, vorgestellt. Dabei werden die Beweggründe und Resultate kurz erläutert. Dieses Kapitel liefert die theoretischen Inhalte für die Implementierung in Kapitel 5.

In Kapitel 3 wird auf die verschiedenen Technologien und Methoden, für die Entwicklung von Touch-Sensoren und der Gestenerkennung, eingegangen.

Das Kapitel 4 beschäftigt sich mit der Analyse unterschiedlicher Multi-Touch Sensoren und deren Einsetzbarkeit für die zugrunde liegenden Konzepte. Dabei werden im ersten Teil Anforderungen definiert, welche in den folgenden Teilen pro Technologie geprüft werden. Abgeschlossen wird mit der Entscheidung, welche Technologie für die Implementierung in Kapitel 5 verwendet wird.

In Kapitel 5 werden die zuvor erwähnten Konzepte von Sommersguter und Moser an die Rahmenbedingungen des Multi-Touch Sensors adaptiert und zu einem Konzept vereint. Im Anschluss erfolgt die Beschreibung der Implementierung dieses Konzeptes für die gewählte Technologie.

Nach der erfolgreichen Implementierung wird diese in Kapitel 6 mit blinden Personen getestet und evaluiert. Das Kapitel schließt mit einem Fazit über den möglichen Praxiseinsatz des in dieser Diplomarbeit entstandenen Prototypen ab.

Zum Abschluss der Diplomarbeit wird in Kapitel 7 die Zukunft dieses Projektes und weitere Vorgehensweisen erläutert.

KAPITEL 2

# Ausgangspunkt

Dieses Kapitel beschäftigt sich mit dem Ausgangspunkt des Projektes und dessen Umfeld. Dabei werden in Punkt 2.1 alle relevanten Projekte aufgezeigt, die über die Jahre auf der TU Wien durchgeführt wurden. Der Abschnitt 2.2 gibt eine Übersicht über alle verfassten Diplomarbeiten zu diesem Thema, deren Ergebnisse für das Konzept im Kapitel 5.1 als Basis verwendet werden. Das Kapitel schließt mit einer Zusammenfassung über die wichtigsten Punkte.

## 2.1 Durchgeführte Projekte

### 2.1.1 Sparkling Fingers

Im Rahmen des Forschungsprojektes Sparkling Fingers[Buna] wurde im Zeitraum von November 2008 bis August 2010 die Grundidee dieses Projektes geboren. Ziel war es ein haptisches E-Learning Instrument für sehbehinderte SchülerInnen zu entwickeln. Mit diesem Instrument sollen zwei- bis dreidimensionale Modelle haptisch erkundet werden können und bei Erreichen diverser Hotspots, Audiodaten wie zum Beispiel Kommentare wiedergegeben werden. Für die Realisierung wurden mehrere Webcams, ein Computer sowie eine Auflagefläche für das zu erkundende Modell verwendet. Be

### 2.1.2 Sparkling Fingers 2.0

Nach dem erfolgreichem Abschluss des Projektes Sparkling Fingers wurde ein Folgeprojekt unter dem Namen Sparkling Fingers 2.0[Bunb] gestartet. Im Zeitraum dieses zweijährigen Projektes (Laufzeit November 2010 bis Dezember 2012) wurde eine E-learning Plattform für sehbehinderte Personen unter der Verwendung von Web 2.0 Aspekte entwickelt. Ziel dieser Plattform ist es ein transportables Tablet mit einer tastbaren Landkarte zu kombinieren. Bei der Interaktion mit dieser Karte sollen bei definierten Hot Spots geolokalisierte Audioaufnahmen abgespielt werden. Ergebnisse dieses Projektes sollen

später in der Sonderpädagogik eingesetzt werden. Das Bestreben ist es, die räumliche Vorstellungskraft bei blinden SchülerInnen, mittels Lesens und Ertastens einer interaktiven Landkarte aktiv anzuregen.

Durchgeführt wurde dieses Projekt an der Technische Universität Wien (TU Wien) in Kooperation mit dem Bundes-Blindenerziehungsinstitut sowie dem SZU - Schulzentrum Ungergasse. An der TU Wien wurden im Rahmen dieses Forschungsprojektes zwei Diplomarbeiten verfasst um diverse wissenschaftliche Hintergründe zu beleuchten. Diese werden in Kapitel 2.2 vorgestellt. Finanziert wurde das Projekt durch das Bundesministerium für Wissenschaft und Forschung (BMWF).

Im Laufe des Projektes sind ein ausgereiftes Konzept und diverse Software- und Hardwareprototypen entstanden.

### 2.1.3 CEAT

Das Projekt Collaboratively Editable Audio Tangibles kurz CEAT ist ein Teilprojekt von Sparkling Fingers 2.0. In diesem Projekt wurde der Fokus auf die Entwicklung eines offenen kapazitiven Multi-Touch Sensors und die Erstellung von taktilen Karten gelegt. Während der Entwicklungslaufzeit wurde der erste Prototyp eines Sensors gebaut, der als Grundlage für diese Diplomarbeit dient. Die Entwicklung wurde durch Armin Wagner und Georg Kaindl geleitet und durchgeführt.

Nach dem Ende der Finanzierung durch das BMWF wird die Entwicklung verbesserter Prototypen bis zur Fertigstellung eines marktreifen Sensors unter dem Namen “Wiretouch“[1] weitergeführt.

## 2.2 Projektbezogene Diplomarbeiten

### 2.2.1 Konzept, Gestaltung und prototypenhafte Implementierung eines Multimedia Authoring Systems für Blinde

Im Rahmen der Diplomarbeit *Konzept, Gestaltung und prototypenhafte Implementierung eines Multimedia Authoring Systems für Blinde* [Mos12] wurden zwei Hardware- und Softwareprototypen entwickelt. Diese Prototypen basieren auf einem Konzept, das mit dem Kernteam erarbeitet wurde. Für die Entwicklung des Prototypen wurde Holz als Werkstoff verwendet. Hierdurch wurde ein erster Einblick ermöglicht, wie die Hardware in Realität aussehen kann und welche Probleme sich bei blinden Personen im Umgang damit ergeben können. Der Softwareprototyp liefert ein Audio gestütztes User-Interface, womit die Anwendungsebene des Projektes Sparkling Fingers 2.0 erstmals abgebildet wurde.

Im Konkreten sind die folgenden Punkte die Kernelemente des Konzeptes, die auch teilweise im Konzept des Kapitels 5.1 als Basis wiederverwendet wurden.

[1] http://www.wiretouch.net

#### Kartenerkennung

Der Applikation muss bekannt sein, welche haptische Karte auf dem Sensor liegt um dem User die richtigen Informationen zur Verfügung zu stellen. Jede Karte besitzt einen unterschiedlichen Kartenausschnitt sowie ein Zoom-Level wodurch sich die dargestellten Koordinaten sehr stark unterscheiden kann. Im Idealfall soll die Karte automatisch durch den Sensor erkannt werden, um die Usability deutlich zu erhöhen. Für die Automatisierung hat Moser zwei Varianten in seiner Diplomarbeit vorgestellt.

- **Optisch:** Die Erkennung der Karte erfolgt mittels einer am Sensor verbauten Kamera, die einen aufgedruckten Code oder ein Muster auf der Karte erkennt und der Applikation mitteilt.
- **RFID:** Durch das Anbringen von RFID Chips auf der Karte und einem dazu passenden Lesegerät im Sensor, ist eine Erkennung der Karte möglich. Dieses Verfahren ist sehr zuverlässig, benötigt aber zusätzliche Soft- und Hardware.

Bei einer manuellen Erkennung der Karte ist die Eingabe eines Codes eine mögliche Variante. Durch das Anbringen des Codes in Braille-Schrift könnte diese am Sensor eingegeben werden. Notwendig wären dafür die passende Hardware für die Eingabe am Sensor.

#### Sprachausgabe

Im Konzept von Moser erfolgt die Sprachausgabe über Kopfhörer oder einem verbauten Lautsprecher. Die Geschwindigkeit der Sprachausgabe und Lautstärke soll dabei über Schieberegler oder ähnliche Konzepte regulierbar sein.

#### Datenhaltung

In der Endausbaustufe sollen die Daten über Webservices an die Software angebunden sein. Die Informationen der Karten werden in einer Datenbank gespeichert und verwaltet. Im Zuge des Prototypen wurden die Webservices durch XML-Dateien ersetzt. Für die ortsbezogenen Daten wird das Koordinatensystem herangezogen, in dem die Orte als Kreise oder Polygone definiert und in der Datenbank mit den Informationen hinterlegt sind. Die Audioinformationen sind als MP3-Dateien verfügbar und werden bei Bedarf von der Software abgespielt.

### 2.2.2 Interaction Design for the Blind

Paul Sommersguter stellt in seiner Master Thesis *Interaction Design for the Blind* [Som13] ein Interaktionskonzept vor, das die Zusammenarbeit der einzelnen Komponenten des Sparkling Fingers 2.0 beschreibt. Ziel war es ein möglichst intuitives und komfortables Gesamtkonzept zu entwickeln, um blinde Kinder beim Kartenlesen zu unterstützen. Seine Arbeit unterteilt sich in zwei Bereiche. Im theoretischen Teil erläutert Paul Sommersguter

relevante Literatur zu Themen wie ***Informationen in Karten darstellen, Multi-Touch und Gesteninteraktion sowie Audio Feedback Design***. Im praktischen Teil beschreibt Sommersguter das Zusammenspiel des Multi-Touch Sensors mit den taktilen Papierkarten und erarbeitet ein Set aus Gesten, um mit diesen zu interagieren. Dieses Konzept basiert auf Kartenleseworkshops und Feedbackgespräche mit blinden Kindern und deren Lehrenden.

Die Kernelemente des Interaktionskonzeptes, die sich auch teilweise in diesem Konzept wiederspiegeln, werden in den folgenden Abschnitten überblicksmäßig vorgestellt.

#### Applikationszustände (States)

Sommersguter beschreibt in seinem Konzept eine Reihe von unterschiedlichen States, die einzelne Zustände der Applikation darstellen. Diese Zustände können auch als Menüpunkte ausgelegt werden, zwischen denen der/die BenutzerIn wählen kann.

- **Erkundungsmodus:** Der Erkundungsmodus wird gestartet, wenn eine Karte auf den Sensor gelegt wird. Beendet wird dieser, wenn die Karte wieder entfernt wird. In diesem Modus kann die benutzende Person die Karte mit seinen Händen erkunden und sich bei Bedarf unterschiedliche Informationen zu den gewählten Punkten ausgeben lassen.
- **Ebenen:** Eine Ebene beinhaltet eine Reihe von geobasierte Informationen. Diese Informationen werden nach Themen gruppiert und jeweils einer Ebene zugeteilt. Eine Ebene beinhaltet alle zu einem Thema zugehörigen Informationen. Die Themen können zum Beispiel wie folgt gruppiert sein: Straßennamen, Parks, Kaffeehäuser und Öffentliche Gebäude. Der/die BenutzerIn kann wählen, welche Ebenen er für die Ausgabe der Informationen aktivieren oder deaktivieren möchte. Sommersguter unterscheidet zwischen drei unterschiedlichen Tiefen von Ebenen: Alle Ebenen, Selektierte Ebenen für diese Erkundungssessions, Aktivierte Ebenen. *Aktivierte Ebenen* ist eine Liste mit Ebenen die für die aktuelle Ausgabe der Audioinformationen berücksichtigt wird. *Selektierte Ebenen* beinhaltet alle Ebenen die für diese Karte ausgewählt wurden und aktiviert weren können (werden dabei der Liste *aktivierte Ebenen* hinzugefügt) oder deaktiviert (werden dabei der Liste *aktivierte Ebenen* entfernt) werden. *Alle Ebenen* beinhaltet alle in der Applikation verfügbaren Ebenen.
- **Ebenen Selektion:** In diesem Modus können die gewünschten Ebenen für die Ausgabe der Informationen gewählt werden. Nur Informationen aus den aktivierten Ebenen werden dem/der BenutzerIn zur Verfügung gestellt. Zusätzlich kann in diesem Modus durch die Einträge (Orte) der einzelnen Ebenen geblättert werden.
- **Orte:** Ein Ort liefert Informationen aus Online Webservices zu einem gewählten Punkt auf der Karte. Dabei unterscheidet Sommersguter zwischen Punkten und Flächen, die auf einer Karte Informationen beinhalten können. Ein Punkt entspricht

einem Kreis, während Flächen einen Bereich als Polygon darstellt. Eine Restaurant kann so zum Beispiel als Punkt dargestellt werden, während ein Park oder Fluss als Fläche auf der Karte hinterlegt wird. Nur hinterlegte Flächen und Punkte die mindestens eine Ebenen-Information haben, können für die Ausgabe von Informationen selektiert werden.

- **Eigene Orte:** Dieser Modus erlaubt es dem/der BenutzerIn eigene Inhalte zu hinterlegen. Zuerst wird ein Punkt oder Bereich auf der Karte ausgewählt, um im zweiten Schritt einen Text oder Audio zu hinterlegen.
- **Gespeicherte Orte:** Bei der Erkundung von Orten kann der/die BenutzerIn sich für ihn wichtige Orte speichern. Diese können in diesem Modus aufgerufen und durchgeblättert werden.
- **Distanz:** In diesem Modus kann die Distanz zwischen zwei gewählten Punkten berechnet werden.
- **Einstellungen:** In den Einstellungen hat der/die BenutzerIn zwei Möglichkeiten. Es kann die Liste der selektierten Ebenen definiert werden, wobei hier standardmäßig alle Ebenen selektiert sind. Des Weiteren können gespeicherte Orte aufgerufen und die dazugehörigen Informationen ausgegeben werden. Diese Liste kann hier gespeicherte Orte unterschiedlicher Karten beinhalten.
- **Hilfe:** Ausgabe von Informationen über die Applikation oder den Sensor.

#### Gesten

Sommerguters Konzept sieht sechs Basis-Gesten vor, die von Saffer[Saf08] wie folgt definiert sind:

- **Tap:** Ein kurzes Berühren der Sensoroberfläche (<100ms).
- **Double Tap:** Zweifache Ausübung eines Tap in einem sehr kurzen Abstand (<75ms).
- **Tap & Hold:** Ein Finger berührt die Sensoroberfläche für einen längeren Zeitraum. Der Finger bewegt sich dabei nicht. Die Aktion endet, wenn der Kontakt verloren geht.
- **Swipe:** Fünf Finger bewegen sich in die gleiche Richtung und führen damit eine Wischgeste aus. Diese Geste kann nach links, rechts, oben und unten ausgeführt werden. Die Geste endet, wenn der Kontakt verloren geht.
- **Pinch:** Zwei von einander entfernte Finger bewegen sich auf einander zu.
- **Drag:** Ein Finger fährt über den Sensor ohne den Kontakt zu verlieren. Die Aktion endet, wenn der Kontakt verloren geht.

Die Gesten Tap, Double Tap, Tap & Hold und Pinch sind Richtungsunabhängig, während die restlichen zwei Gesten Swipe und Drag in unterschiedlichen Richtungen ausgeführt werden können und somit eine jeweils andere Bedeutung haben.

Um dem/der BenutzerIn eine möglichst einfache und vielseitige Interaktionsmöglichkeit mit der Applikation zu bieten, ist es auch möglich die oben genannten Gesten mit mehr als den in der Definition beschriebenen Fingern auszuführen. Dadurch erweitern sich die Gesten um folgende Varianten:

- **Tap (1)/(2):** Tap mit einem und zwei Fingern.
- **Double Tap (1):** Double Tap mit einem Finger.
- **Tap & Hold (1)/(2):** Tap & Hold mit einem und zwei Fingern.
- **Swipe (5):** Einen Swipe mit fünf Fingern.
- **Pinch Open (5):** Einen Pinch mit fünf Fingern die sich von einander wegbewegen[2].
- **Drag (1):** Drag mit einem Finger.

### Zustandsdiagramm

Das Zustandsdiagramm gibt eine grafische Übersicht über alle zuvor definierten Zustände und deren Übergänge. Zusätzlich werden alle möglichen Gesten pro Zustand angeführt. Die Abbildung 2.1 zeigt das Zustandsdiagramm, wie es als Endresultat der Diplomarbeit von Sommersguter präsentiert wurde.

Der Einstiegspunkt der Applikation ist der Zustand *Start*. In diesem Zustand ist die Karte noch nicht auf dem Sensor platziert. Ab diesem Zeitpunkt können die *Einstellungen (Settings)* oder die *Hilfe (Help)* aufgerufen werden. Durch das Auflegen der Karte erfolgt der Übergang in den *Entdeckungsmodus (Silence)*. Durch die Auswahl eines Punktes auf der Karte mit der Geste Tap & Hold (1) werden die dazugehörigen Informationen im Zustand *Ort (Location)* ausgegeben. Durch das Ausführen einer Pinch Open (5)-Geste, wird der Zustand *Ebenen (Layers)* gewählt. Das Berechnen einer *Distanz (Distance)* kann durch Tap & Hold mit zwei Fingern durchgeführt werden.

Die beschriebenen Gesten bei den Zuständen in der Grafik geben an, welche Gesten zur Navigation im jeweiligen Zustand durchgeführt werden können.

[2]Diese Geste wird in der Definition von Saffer[Saf08] auch Spread genannt. Um die gleichen Bezeichnungen wie bei der Diplomarbeit von Sommersguter[Som13] zu verwenden wird der Name “Pinch Open“beibehalten.

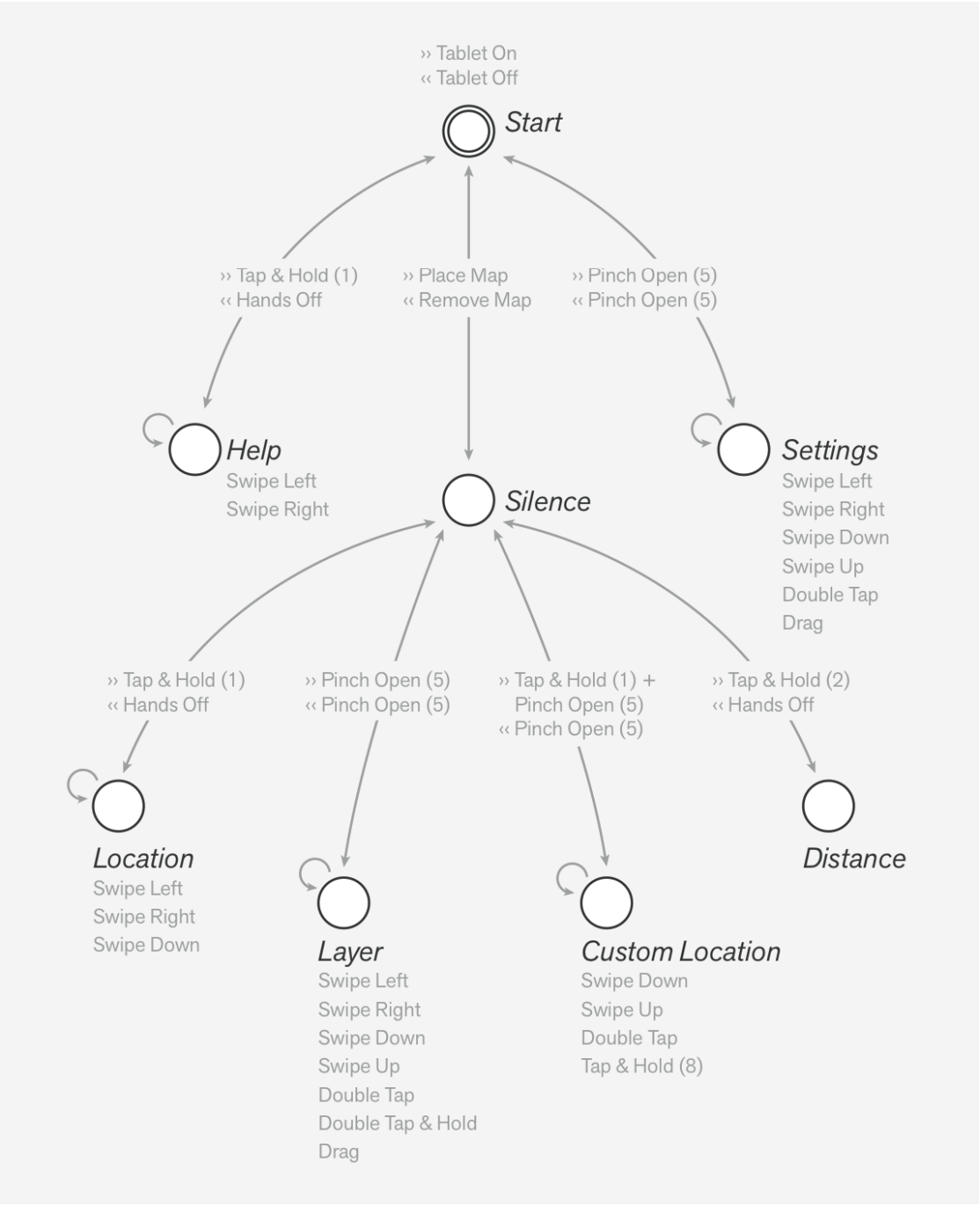


Abbildung 2.1: Dieses Zustandsdiagramm beschreibt alle vorgesehen Zustände des Konzeptes von Sommersguter, sowie die möglichen Gesten die in diesem Zustand ausgeführt werden können. Die beschriebenen Gesten unterhalb der Zustandsbezeichnungen können in den jeweiligen Zuständen zur Navigation ausgeführt werden. [Som13].

## 2.3 Zusammenfassung

Im Rahmen unterschiedlichster Projekte und Arbeiten wurden bereits zahlreiche Ideen erarbeitet und Konzepte getestet. Auch die bisher implementierten Prototypen decken einige Aspekte ab und lieferten neue Erkenntnisse über die Verwendbarkeit und den Umgang mit Technologien zur Verfolgung dieser Konzepte. Trotz der vorangeschrittenen Zeit seit der letzten Publikation von Forschungsergebnisse im Jahr 2013 gibt es bis heute noch keinen vollständigen Prototypen, der alle Anforderungen abdeckt. Dieser Prototyp soll nun im Rahmen dieser Diplomarbeit entstehen.

Sowohl Moser als auch Sommersguter liefern mit ihren Überlegungen und Konzepten zahlreiche Ansätze, die sich in dieser Diplomarbeit wieder finden. Das kombinierte und um eigene Ideen und Verbesserungen erweiterte Konzept ist im Kapitel 5.1 zu finden.

KAPITEL 3

# Stand der Forschung und Technik

Dieses Kapitel beschäftigt sich mit dem aktuellen Stand der Forschung in Themenbereichen, die für diese Diplomarbeit relevant sind. Im ersten Abschnitt des Kapitels werden Projekte vorgestellt, welche durch den Einsatz von Technik versuchen, Beeinträchtigungen blinder Personen auszugleichen. Die Grundidee dieser Diplomarbeit basiert auf dem Einsatz von Sensoren, die Interaktionen durch Finger und Gesten erkennen können. Im Abschnitt 3.2 dieses Kapitels werden aktuelle, auf dem Markt vorhandene, Multi-Touch Sensoren vorgestellt. Im Zuge dieser Vorstellung werden die unterschiedlichen Technologien näher betrachtet und beschrieben. Abschließend werden die wichtigsten vorgestellten Projekte und Technologien zusammengefasst.

## 3.1 Projekte

Dieser Abschnitt soll weitere Projekte aufzeigen, die ähnliche Konzepte und Ideen verfolgen, um Sehbeeinträchtigungen auszugleichen. Ziel ist es, ähnliche Herausforderungen und deren Lösungsansätze zu erkennen und das Wissen in der Realisierung des Konzeptes und der Erstellung des Prototypen in Kapitel 5 einfließen zu lassen.

### 3.1.1 TouchOver Karten

Durch die große Verbreitung von Smartphones und der vielseitigen Einsatzzwecke gibt es kaum Informationen, die nicht stetig über das Internet abrufbar sind. Ein häufiger Einsatz ist die Erkundung und Navigation durch die Verwendung von GPS und diverser Kartenapplikationen am Mobiltelefon. Das Projekt “TouchOver Map“ von Poppinga et al[PMPRG11] zeigt eine Variante auf, mit der blinde Personen diese Geräte ebenfalls für die Erkundung der Umgebung verwenden können. Das Konzept setzt dabei auf die Ausgabe von Sprache als Audio und haptischen Feedback durch Vibration, um dem/der BenutzerIn Informationen zu einzelnen Punkten auf der Karte zu geben. In

dieser Applikation werden die Namen der Straßen, beim Berühren dieser am Bildschirm, mittels Sprache ausgegeben. Die Vibration signalisiert dem/der BenutzerIn, dass der erkundende Finger sich auf einer Straße befindet. Als Gerät wird ein handelsübliches Smartphone mit einer speziell entwickelten Kartenapplikation eingesetzt.

Der Ansatz, dass bei der Bewegung des Fingers über gewisse Punkte Informationen ausgegeben werden, wird auch von einigen weiteren Projekten verfolgt. Carroll et al[CCL13] erstellten eine Wetterkarte, die den blinden BenutzerInnen Informationen über den selektierten Ort und die Temperatur als Ton in einer bestimmten Höhe ausgibt.

### 3.1.2 Aufnahme von Routen

Um blinde Personen bei ihrer Fortbewegung im Freien zu unterstützen, entwickelten Jafri et al[JA14] ein Konzept für eine Applikation, welche es ermöglicht, Routen und Gehwege aufzuzeichnen. Diese gespeicherten Routen können zu einem späteren Zeitpunkt zur Navigation dieser Wege verwendet werden. Das Hauptaugenmerk liegt dabei auf einer preiswerten, portablen und einfach zu bedienenden Lösung. Im Vergleich zu herkömmlichen Navigationsapplikationen können in dieser Variante Wege aufgezeichnet werden, die nicht auf Karten hinterlegt sind, wie zum Beispiel ein kleiner Weg durch einen Park oder über eine Wiese.

### 3.1.3 Navigation in Einkaufsparks

In zahlreichen Städten gibt es Einkaufsparks, die sich über große Flächen mit einer Vielzahl an Geschäften und Gebäuden erstrecken. Im Gegensatz zu einem Einkaufszentrum sind diese Geschäfte nicht in einem großen Gebäude angesiedelt, sondern nebeneinander angeordnet und nur über Wege im Freien erreichbar. Sich ohne Vorwissen in solchen Parks zurechtzufinden, ist meist nur durch die Verwendung von Plänen und dem Folgen einer Beschilderung möglich. Personen mit einer Sehbeeinträchtigung sind hier auf zusätzliche Informationen angewiesen. Paladugu et al[PCZL13] befassten sich in ihrer Publikation mit dem Versuch, diese Art von Informationen für blinde Personen zur Verfügung zu stellen. Sie beschäftigen sich dabei mit der Herkunft der Daten, sowie dem Zusammenfassen der Informationen aus unterschiedlichen Quellen. Im Anschluss wird eine iPhone-Applikation vorgestellt, welche die aufbereiteten Daten über eine Sprachausgabe dem/der BenutzerIn ausgibt.

### 3.1.4 Brainstorming mit Mind Maps

Sehr häufig werden für das Sammeln von Ideen oder das Treffen von Entscheidungen Mind Maps als Hilfe verwendet. Diese können sowohl für Einzelpersonen, aber auch in der Gruppe zur schnellen Informationssammlung führen. Dabei werden die Informationen in einer Baumstruktur gesammelt und gruppiert nach Themen abgelegt. Diese Tools setzen dabei sehr stark auf die Visualisierung der Inhalte und sind für die Verwendung durch blinde Personen nicht beziehungsweise weniger geeignet. Walker et al[SWAO$^+$14] beschreiben einen Ansatz für ein kollaboratives Mind Map-Tool namens CoME (Collaborative Mind

Map Editor), das für eine Gruppe von Personen mit und ohne Sehbeeinträchtigungen verwendbar ist. Dazu gehören Lösungsansätze für die Sprachausgabe von Elementen in der Mind Map, sowie die Unterstützung der Ausgabe von non-verbalen Kommunikationselementen. Dabei werden Technologien wie Leap Motion Sensoren für die Erkennung von Gesten und ein Microsoft PixelSense zur Darstellung der Mind Map verwendet. In dieser Arbeit wird dabei auf bereits entwickelte Konzepte, wie zum Beispiel der Darstellung der Inhalte für blinde Personen, von Pölzer et al[PSWP13] zurückgegriffen.

## 3.2 Multi-Touch Devices

Die Forschung zum Thema der Multi-Touch Oberflächen und Geräte liegt bereits einige Jahrzehnte zurück. Eine der bekanntesten Aufzählungen dieser entwickelten Technologien lieferte Buxton in seiner Publikation "Multi-Touch Systems that I have Known and Loved"[Bil14]. Diese Aufzählung deckt eine breite Palette an Multi-Touch Entwicklungen ab, die in den Jahren von 1965 bis 2011 entwickelt wurden. Vor allem im letzten Jahrzehnt sind zahlreiche Geräte am Verbrauchermarkt erschienen, die als Basis für eine Vielzahl an neuen Projekten verwendet werden.

In diesem Abschnitt werden die wichtigsten Funktionsweisen der Multi-Touch Geräte vorgestellt und im Anschluss die aktuell am Verbrauchermarkt erhältliche Geräte präsentiert. Es werden vor allem Technologien betrachtet, welche für eine Analyse der Erstellung eines Prototypen in Kapitel 4 in Frage kommen.

### 3.2.1 Grundlegende Technologien

Schöning et al[SBD08] beschreiben in ihrer Publikation unterschiedliche Funktionsweisen von Multi-Touch Technologien. Diese decken nach wie vor den Großteil der zur Zeit verfügbaren Geräten ab. Im folgenden Abschnitt werden diese Varianten in einem kurzen Überblick vorgestellt. Detailliertere Informationen zu diesen Technologien können der Publikation von Schöning et al[SBD08] sowie den unterhalb angeführten Referenzen entnommen werden.

#### Optische Ansätze

Für den optischen Ansatz werden Interaktionen durch das Auswerten von Bildern erkannt. Dabei werden Infrarotkameras und infrarotes Licht eingesetzt. Bei einer Berührung der Oberflächen wird infrarotes Licht durch den Touchpunkt reflektiert, sodass es von einer unter der Oberfläche positionierten Infrarotkamera aufgezeichnet werden kann. Diese Bilder werden anschließend durch eine Steuereinheit ausgewertet und verarbeitet. Grundsätzlich wird zwischen dem Ansatz der Totalreflexion (Frustrated Total Internal Reflection (FTIR)) und jenem des diffusen Lichtes (DI) unterschieden.

Durch die relativ einfache Bauweise werden diese Ansätze gerne bei do-it-yourself Projekten eingesetzt. Alle dafür notwendigen Bauteile sind einfach und kostengünstig zu

erwerben und können händisch zusammengebaut werden[SHBS10, Han05]. Kommerzielle Lösungen werden, vor allem in Form von Tabletop-Geräten, zum Beispiel für Übersichtskarten in Einkaufszentren und bei Messen angeboten[Str10, MTF10].

#### Widerstand basierte Ansätze

Die Funktionsweise der auf Widerstand basierten Touch-Oberflächen besteht aus zwei leitenden Schichten, die durch Abstandshalter voneinander getrennt sind. Die Frontplatte besteht dabei aus einem flexiblen, hart-beschichteten Material. Bei einem Touch auf dieser Oberfläche findet eine stellenweise Berührung der zwei darunterliegenden leitenden Schichten statt. Durch die Messung des elektrischen Widerstandes und der auf den Leiterplatten angelegten Spannung, kann die Position des Touches berechnet werden.

Diese Methode besitzt einen sehr geringen Stromverbrauch und eignet sich besonders gut für den Einsatz in mobilen Gräten, wie zum Beispiel Digitalkameras. Die Berührungspunkte können sowohl durch den Einsatz von Fingern, als auch Stifte erkannt werden.

#### Kapazitive Ansätze

Bei den kapazitiven Ansätzen wird zwischen Oberflächen-kapazitive und Projiziert-kapazitive Touchscreens unterschieden.

Bei der Verwendung eines Oberflächen-kapazitiven Touchscreens wird eine Glasplatte mit einer leitenden Folie beschichtet. Durch das Erzeugen eines gleichmäßig elektrischen Feldes an den vier Ecken der Folie, können Touchpunkte durch die Erzeugung von kleinen elektrischen Strömen bei der Berührung der Folie gemessen werden.

Bei einem Projiziert-kapazitiven Touchscreen werden zwei Ebenen, mit einem zueinander gegenteiligen leitbaren Muster (zum Beispiel mit horizontalen und vertikalen Linien im Schachbrettmuster), übereinandergelegt und durch eine dünne isolierte Schicht von einander getrennt. Während eine Ebene für die Messung zuständig ist, wird die zweite Ebene mit einer Spannung belegt. Durch eine Berührung auf der Oberfläche kann der Spannungsabfall gemessen, und daraus die Position des Touchpunktes berechnen werden. Im Vergleich zur Oberflächen-kapazitiven Methode, können durch diesen Ansatz mehrere Touchpunkte einfacher und zuverlässiger erkannt werden[SBD08].

#### Alternative Ansätze

Zusätzlich zu diesen oben genannten Funktionsweisen wurde im Laufe der Erstellung dieser Diplomarbeit ein Gerät veröffentlicht, das einen alternativen Ansatz verwendet. Durch den Einsatz von druckempfindlichen Sensoren werden die Touchpunkte am Sensor erkannt. Dieser Ansatz wird im nächsten Abschnitt “Aktuelle Hardware“ beim Produkt Sensel Morph näher vorgestellt.

### 3.2.2 Aktuelle Hardware

Durch die in der Zwischenzeit hohe Zahl an Geräten, die Multi-Touch unterstützen, kann im Zuge der Diplomarbeit lediglich eine Auswahl von Geräten näher vorgestellt werden. Für diese wurden Geräte mit unterschiedliche Ansätzen herangezogen, die für die Verwendung als Prototyp dieser Diplomarbeit in Frage kommen.

#### Apple iPad Pro

Das Apple iPad Pro wurde im Herbst 2015 von der Firma Apple veröffentlicht. Es ist mit 12,9 Zoll das bisher größte Produkt seiner Serie und bietet eine kapazitive Multi-Touch Fläche von 26,2 x 19,7cm. Mit einer Auflösung von 2732 x 2048 Pixel bei 264PPI ist es ebenfalls das hochauflösendste der iPad-Serie. Das Tablet ist mit einem A9X-Chip mit 64-Bit Architektur und einem M9 Motion Coprozessor ausgestattet. Zusätzlich verfügt das iPad Pro über eine Kamera auf der Vorder- und Rückseite. Das Gerät besitzt vier Lautsprecher für die Audioausgabe und zwei Mikrofone für Anrufe und Audioaufnahmen[Appc].

Im Frühling 2016 wurde eine kleinere Variante, das iPad Pro mit 9,7", präsentiert.

Abbildung 3.1: Das Apple iPad Pro (Late 2015) mit einem kapazitiven 12,9"Multi-Touch Display[Appc].

Das iPad Pro wird in Kapitel 4 als Technologie, für den Einsatz des zugrunde liegenden Konzeptes, näher vorgestellt und evaluiert.

### Microsoft Surface

Microsoft bietet aktuell drei unterschiedliche Varianten an Multi-Touch Devices: Surface Book, Surface Pro 4 und das Surface 3[Mic]. Während das Surface 3 oder Surface Pro 4 Tablets sind, stellt das Surface Book eine Mischung aus einem Tablet und einem Notebook dar. Durch die abnehmbare Tastatur lässt es sich wie ein Tablet verwenden, während es mit der Tastatur zu einem vollwertigen Laptop wird.

- **Surface Book:** Das Surface Book besteht aus einem kapazitiven 13,5"-PixelSense™Multi-Touch Display. Die Auflösung des Bildschirms beträgt 3000 x 2000 Pixel bei 267PPI und besitzt damit das größte und bestauflösendste Display der Surface Serie.
- **Surface Pro 4:** Das Surface Pro 4 ist das zweit größte Display der Surface Serie mit einem 12,3"-PixelSense™Bildschirm und einer Auflösung von 2736 x 1824 Pixel bei 267PPI.
- **Surface 3:** Das Surface 3 ist das kleinste Tablet mit einer Display Größe von 10,3". Die Auflösung beträgt dabei 1920 x 1280 Pixel bei 214PPI.

Die genannten Geräte sind auf der Front- und Rückseite mit einer Kamera für die Aufnahme von Videos und Fotos ausgestattet.


Abbildung 3.2: Microsoft Surface Serie (Stand April 2016). Von links nach rechts: Surface Pro 4, Surface Book und das Surface 3[Mic].

### Sensel Morph

Sensel Morph[Sen] ist ein auf Druck reagierendes Touch-Eingabegerät. Die Touchoberfläche besteht aus 20.000 Drucksensoren, die auf einer Fläche von 24.0 x 16.9 cm im Abstand von 1.25mm angebracht sind. Die gemessenen Werte der Sensoren werden in hochauflösende Bilder umgewandelt und verarbeitet. Dabei werden alle Touchpunkte und Druckwerte ausgewertet. Die Abfrage der Werte erfolgt mit einer Frequenz von >125HZ und einer maximalen Latenzzeit von 8ms.

Im Vergleich zu Technologien, die nur leitfähige Objekte erkennen können, erkennt der Sensel Morph alle Objekte, unabhängig der Leitfähigkeit, durch die Verwendung von

Drucksensoren. Diese Methode ermöglicht es unterschiedliche Belege auf den Sensor zu legen und Touchpunkte auf diesen Belegen zu erkennen. Sensel sieht für das Gerät standardmäßig sechs unterschiedliche Belege vor, die je nach Anwendungsgebiet eingesetzt werden können. Diese Belege reichen von klassischen Tastaturen, über Tasten eines Klaviers bis hin zu Knöpfen und Regler für die individuelle Belegung mit Funktionen. Der Wechsel der Belege erfolgt über Magnete und wird durch eindeutige in den Magneten verbaute Codes automatisch erkannt.

Abbildung 3.3: Das Sensel Morph ist ein auf Druck reagierendes Touch-Eingabegerät mit 20.000 Drucksensoren im Abstand von 1.25mm. In dieser Darstellung ist das Sensel Morph via Bluetooth mit einem Tablet verbunden. Die Ausgabe am Tablet zeigt die erkannten Druckpunkte des Sensors.[Sen]

Im Vergleich zu Tablets besitzt das Sensel Morph keinen eigenen Ausgabe Bildschirm und benötigt dafür eine Verbindung zu einem PC oder mobilen Gerät. Dabei werden Linux, Mac OSx, Windows 7/8/10 und iOS als Betriebssystem unterstützt.

Das Projekt Sensel Morph wurde im August 2015 über die Internetplattform Kickstarter[1]

[1] https://www.kickstarter.com/projects/1152958674/the-sensel-morph-interaction-evolved

zur Finanzierung vorgestellt. Zum Zeitpunkt der Erstellung der Diplomarbeit war Sensel Morph noch nicht lieferbar. Eine Analyse des Sensel Morphs war daher im Rahmen der vorliegenden Diplomarbeit nicht möglich.

## 3.3 Gestenerkennungs Sensoren

Zusätzlich zu den vorgestellten Multi-Touch Geräten gibt es eine Reihe an Sensoren, die sich dem Thema Gestenerkennung angenommen haben, ohne dabei direkt mit Displays zu interagieren. Dieser Abschnitt stellt einige dieser Technologien vor. Einer der Sensoren wird in Kapitel 4 für den Einsatz des im Rahmen dieser Diplomarbeit entstehenden Prototypen evaluiert.

### 3.3.1 Leap Motion Controller

Der Leap Motion Controller[Leab] ist ein kleines USB Gerät, das Interaktionen über dem Bereich des Sensors erkennt. Der Sensor ist mit zwei Infrarotkameras und drei Infrarot-LEDs ausgestattet. Bei einer Aktivität über dem Sensor wird das durch die LED Lichter ausgestrahlte Licht durch die Hände oder Gegenstände teilweise reflektiert und durch die Infrarotkamera registriert. Die empfangenen Bilder werden am Controller im lokalen Speicher geladen, verarbeitet und an die Leap Motion Tracking Software via USB weitergeleitet. Jene vom Sensor empfangen Bilder zeigen dabei nur die Gegenstände, die direkt durch die LEDs angestrahlt wurden und das Licht reflektierten. Der Sensor kann mehrere Hände, Finger und Gegenstände (Stifte) erkennen, solange sich diese in dem Bereich über dem Sensor befinden. Die Kamera kann bis zu 200 Frames pro Sekunde verarbeiten, was eine sehr präzise Interpretation der Daten in Echtzeit ermöglicht. Der Sensor hat einen Sichtbereich von 150 Grad und einen Interaktionsraum von ungefähr 226dm$^3$ (60x60x60cm). Dieser startet ungefähr 2,5cm über dem Sensor bis zu einer Höhe von 60cm[Leaa].

Der Leap Motion Controller überzeugt durch seine zuverlässige Erkennung von Fingern beziehungsweise Händen und der günstigen Anschaffung. Durch die geringe Größe und des geringen Gewichtes lässt sich der Controller sehr vielseitig einsetzen. In zahlreichen Projekten wurde die Verwendbarkeit des Leap Motion Sensors für den Einsatz in unterschiedlichen Umgebungen geprüft. Die Bandbreite reicht dabei von Musikanwendungen[HG14, SdAdA13], bis zur Steuerung von Fernsehgeräten[VZ14], Einsatzzwecke in medizinischen Geräten[CFRV15] und kollaborativen Mind Maps[SWAO$^+$14].

Der Leap Motion Controller wird in Kapitel 4 als Technologie für den Einsatz des zugrunde liegenden Konzeptes evaluiert.

### 3.3.2 Myo

Das Myo Armband der Firma Thalmic Labs ist ein kabelloses Armband zum Erkennen von Gesten. Durch das Tragen des Armbandes am Unterarm können die Muskelströme mittels Elektromyografie (EMG) Sensoren ausgelesen und interpretiert werden. Dabei

Abbildung 3.4: Technischer Aufbau des Leap Motion Sensors. Auf der Platine lassen sich dabei die zwei verbauten Infrarotkameras, sowie die drei Infrarot LEDs gut erkennen.[Leab]

können fünf Gesten, sowie die Bewegung und Rotation des Armes erkannt werden. Diese Daten werden via Bluetooth zu einem Computer oder mobilen Gerät übertragen und weiterverarbeitet. Standardmäßig werden die folgenden fünf Gesten erkannt: Wave Links, Wave Rechts, Double Tap, Faust und Finger spreizen.

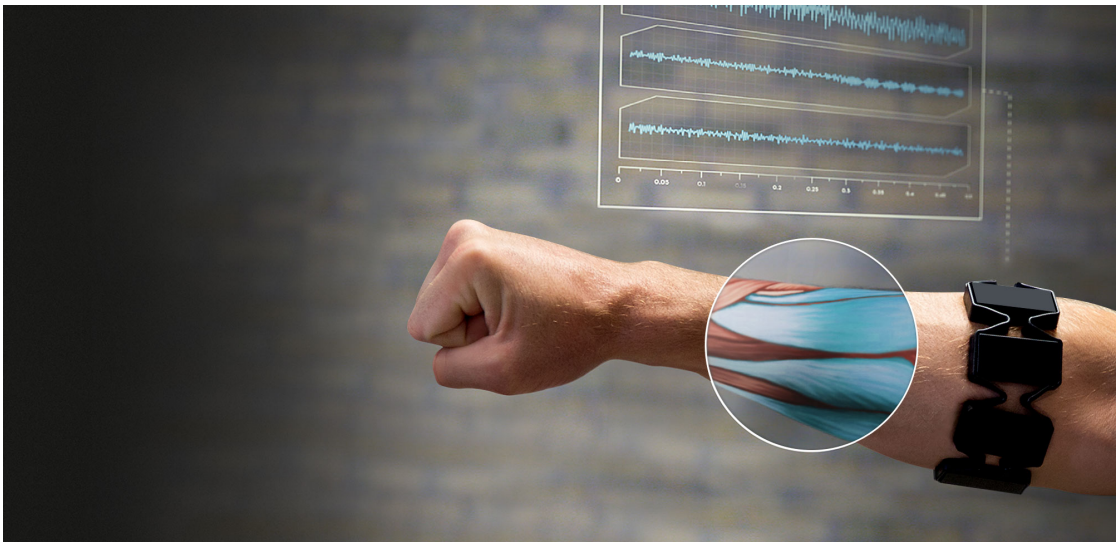

Abbildung 3.5: Das Myo Armband misst Muskelströme mittels Elektromyografie (EMG) Sensoren und interpretiert diese Messdaten als Gesten. Diese Informationen werden zur Verarbeitung via Bluetooth an einen Computer oder ein mobiles Gerät weitergeleitet.[Tha]

## 3.4 Zusammenfassung

Der erste Teil dieses Kapitel widmet sich der Vorstellung von Projekten, die durch den Einsatz von Technologie versuchen, Beeinträchtigungen von blinden Personen auszugleichen. Dabei werden die Problemstellungen und Lösungsansätze von Projekten der Themenbereiche Navigation, Erkundung der Umgebung und der Verwendung von Mind Maps näher beschrieben.

Der zweite Abschnitt erklärt die unterschiedlichen Funktionsweisen von Multi-Touch Geräten und stellt aktuell erhältliche Produkte vor. Dabei liegt der Fokus auf Geräte, die auch für den Einsatz im Rahmen dieser Diplomarbeit in Frage kommen.

KAPITEL 4

# Analyse und Evaluierung von Multi-Touch Sensoren

Dieses Kapitel beschäftigt sich mit der Evaluierung von Multi-Touch Sensoren für die Verwendbarkeit des zu Grunde liegenden Konzeptes von Sommersguter[Som13]. Im Abschnitt 4.1 des Kapitels werden die dafür notwendigen Anforderungen definiert. In den darauffolgenden Bereichen wird der Leap Motion Sensor (Abschnitt 4.2) und das iPad Pro von Apple (Abschnitt 4.3) gegen diese Anforderungen geprüft.

Das Ziel dieses Kapitels ist es, die am besten geeignete Technologie für die Implementierung eines Prototypen zu finden.

## 4.1 Anforderungen

Parallel zur Entwicklung des Konzeptes im Rahmen einer Diplomarbeit, wurde ein Multi-Touch Sensor des Projekt Teams CEAT[Arm13], beziehungsweise später unter dem Namen Wiretouch[Arm14], entwickelt. Dieser Multi-Touch Sensor sollte ein preiswerter do-it-yourself kapazitiver Sensor sein, der die gewünschten Anforderungen abdeckt. Leider entstand bis zum Zeitpunkt dieser Diplomarbeit daraus kein fertiges Produkt, das man für die Implementierung des Prototypen verwenden könnte. Aus diesem Grund wurden im Zuge dieser Arbeit Anforderungen aus dem Konzept von Sommersguter[Som13] abgeleitet, um alternative Technologien zu finden und diese im Bezug darauf zu prüfen. Die meisten der Anforderungen entstehen aus dem Zusammenspiel der Erkennung von Touch-Punkten und den haptischen Karten. Diese Anforderungen sind in den nächsten Abschnitten detailliert beschrieben.

### 4.1.1 Größe des Sensors

Die benutzende Person interagiert mit dem System über eine haptische Karte, indem sie diese mit den Fingern erkundet. Für die gleichzeitige Verwendung beider Hände beziehungsweise aller zehn Finger, muss die Karte mindestens die Größe eines A4-, oder A3-Blattes haben, um genügend Platz für das Erkunden zu bieten. Daraus ergibt sich, dass der Sensor Touches auf einer Fläche von mindestens eines A4 Blattes erkennen muss.

Da diese Lösung von blinden Personen verwendet werden soll, ist es wichtig, dass das Gerät bei der Interaktion mit der taktilen Karte genügend Platz bietet. Dadurch soll gewährleistet sein, dass keine Gegenstände, welche zum Aufbau des Gerätes gehören, Hindernisse für die Erkundung sind und im Weg stehen oder im schlimmsten Fall sogar zerstört werden könnten.

### 4.1.2 Empfindlichkeit

Ist der Sensor unter der haptischen Karte angebracht, muss es möglich sein, die Touch-Punkte auf dem Papier zu erkennen. Die verwendeten taktilen Karten bestehen aus einem dicken Papier (vergleichbar mit einem 160g Papier), auf dem sich Erhebungen mit einer ungefähren Dicke von 0.5mm[GL96] befinden. Der Sensor muss somit in der Lage sein Touches zu erkennen, die nicht direkt auf dessen Sensor platziert werden sondern auf einem Blatt Papier mit einer Dicke von ungefähr 0.6mm.

### 4.1.3 Genauigkeit

Die Interaktion zwischen der Software und dem/der Benutzenden erfolgt über Touch-Punkte und Gesten auf den haptischen Karten. Im Normalfall entspricht ein Touch-Punkt ungefähr $1cm^2$. Der Sensor muss somit in der Lage sein Touches mit einer Genauigkeit von mindestens $1cm^2$ zu erkennen. Abweichungen davon können in der Software Fehlinterpretationen oder falsche Zustände auslösen.

### 4.1.4 Multi-Touch und Gesten Unterstützung

Bei der Konzeptionierung einer Software für sehbeeinträchtigte Menschen ist Benutzerfreundlichkeit ein großer Schwerpunkt. Um eine möglichst einfache und natürliche Interaktion mit der Software zu ermöglichen, soll die Steuerung mittels Gesten erfolgen. Der Sensor muss in der Lage sein sowohl 1-Finger Touch Gesten als auch Multi-Finger Touch Gesten zu erkennen.

### 4.1.5 Physischer Aufbau

Der Aufbau des physischen Prototypen muss einfach und schnell sein, um im Idealfall auch von blinden Personen durchgeführt werden zu können. Es soll gewährleistet sein, dass die Montage und Konfiguration so eingestellt und gespeichert ist, dass keine weitere Aktivität beim wiederholten Benutzen durchgeführt werden muss.

Bei der operativen Verwendung des Gerätes muss sicher gestellt sein, dass die haptischen Karten einfach und schnell gewechselt werden können. Bei der Verwendung der Karten muss darauf geachtet werden, dass diese bei der Verwendung von Gesten nicht auf dem Sensor verrückt.

### 4.1.6 Wartung

Der Sensor soll keine, oder nur geringe Wartungen benötigen, da diese unter Umständen nicht durch sehbeeinträchtigte Menschen alleine durchgeführt werden können.

### 4.1.7 API Schnittstellen

Für die Implementierung eines Prototypen muss der Sensor eine offene API[1] - Schnittstelle zur Verfügung stellen. Diese Schnittstelle sollte im Idealfall in einer gängigen Programmiersprache (wie zum Beispiel: Java, JavaScript, C++) verfügbar sein.

## 4.2 Leap Motion Sensor

Der Leap Motion Sensor[Leab] ist ein kleines USB Gerät das es ermöglicht, Hände in einem Bereich über dem Sensor zu erkennen. Der Leap Motion Sensor verfügt über zwei Infrarotkameras, die das reflektierte Licht der eingebauten drei Infrarot LEDs aufnehmen und verarbeiten. Die Software bereitet diese Informationen im 3-Dimensionalen Raum auf und stellt diese dem Computer zur Verfügung. Der Sensor kann beide Hände, alle 10 Finger und Gegenstände (Stifte) erkennen, solange sich diese in dem Bereich über dem Sensor befinden. Die Kamera kann bis zu 200 Frames pro Sekunde verarbeiten, was eine sehr präzise Interpretation der Daten in Echtzeit ermöglicht.

Der Sensor hat einen Sichtbereich von 150 Grad und einen Interaktionsraum von ungefähr 226dm$^3$ (60x60x60cm). Dieser startet ungefähr 2,5cm über dem Sensor bis zu einer Höhe von 60cm. Eine räumliche Darstellung des Interaktionsraumes ist in der Grafik 4.1 abgebildet.

Die Mindestvoraussetzung für die Verwendung des Leap Motion Sensors ist ein Computer mit Windows 7/8 oder Mac OS X 10.7, mit 2GB RAM und einem USB 2.0 Anschluss. Als Schnittstellen werden APIs in den Programmiersprachen: JavaScript, Unity, C#, C++, Java, Phyton, Objective-C und Unreal angeboten. Für den Zugriff auf die APIs und deren Dokumentation ist ein kostenloses Benutzerkonto als Entwickler erforderlich.

Der Leap Motion Sensor wurde für die Steuerung des Computers beziehungsweise deren Anwendungen konzeptioniert. Ein App Store ermöglicht kostenlose, als auch kommerzielle Apps aus unterschiedlichen Kategorien zu erwerben. Die Bandbreite reicht dabei von einem Spiel, über eine Lernsoftware, bis hin zur Steuerung des PCs. Zum Zeitpunkt der Diplomarbeit waren rund 200 verschiedene Applikationen im App Store vorhanden.

[1] Englisch: Application programming interface - Wörtlich übersetzt: Anwendungsprogrammierschnittstelle

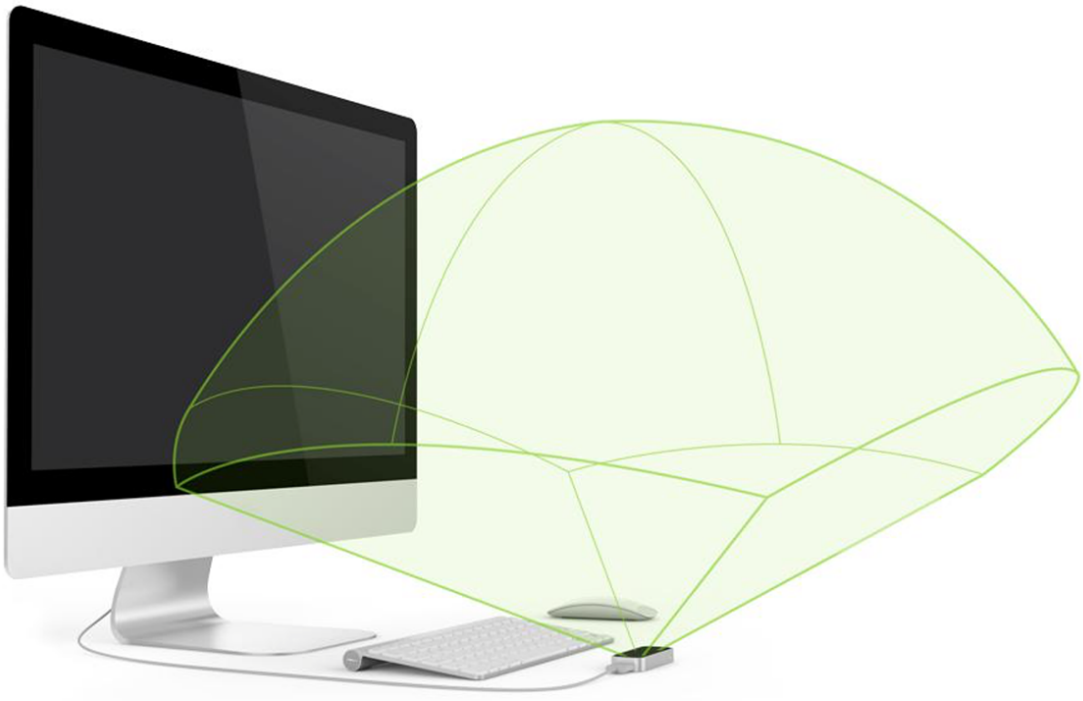

Abbildung 4.1: Darstellung des Interaktionsraumes bei der Verwendung des Leap Motion Sensors[Leab].

### 4.2.1 Vorgehen

Das Vorgehen für die Evaluierung des Leap Motion Sensors besteht aus mehreren Schritten:

- Im ersten Teil 4.2.2 wird eine Demo-Applikation erstellt, welche die Daten des Sensors entgegennimmt und verarbeitet. Das Ziel dieser Applikation ist es im Endstadium Touches auf einer Karte zu erkennen und diese der richtigen Position auf der Karte zu zuordnen. Des Weiteren muss ein Mechanismus entwickelt werden, wie die Karte mit dem Sensor kalibriert werden kann.
- Auf Grund des benötigten Sichtkontaktes zwischen dem Sensor und den Fingern kann der Sensor nicht unter der Karte angebracht werden. Deshalb müssen unterschiedliche Konzepte geprüft werden, wie der Sensor angebracht werden kann. Zusätzlich ist hier eine Prüfung über die Genauigkeiten bei den einzelnen Positionen relevant. Diese Punkte werden im Abschnitt 4.2.3 geprüft.
- Der letzte Schritt (Abschnitt 4.2.4) ist, alle Anforderungen an die Multi-Touch Sensoren zu prüfen. Diese stellen die Basis für die Entscheidung dar, ob dieser Sensor für einen Implementation des Konzeptes geeignet ist.

Da es sich hier um einen ersten Test dieser Technologie handelt, wird nicht zwingend auf einen nachhaltigen Software Quellcode geachtet werden. Im Vorrang steht die Validierung ob der Sensor für die gegebenen Anforderungen geeignet ist.

### 4.2.2 Demo Applikation

#### Vorbereitung und Installation

Die Installation des Leap Motion Sensors ist sehr einfach. Der Sensor wird an dem USB-Anschluss des Computers angebracht und die Software zum Installieren kann von der Herstellerwebsite für Windows oder Mac OS X geladen werden. Nach der kurzen Installation ist der Sensor für die Verwendung bereit.

Für unsere Testzwecke wurde der Sensor an einem Mac Book Pro (Retina, Mid 2012) mit dem Betriebssystem OS X El Capitan (10.11) angeschlossen. Die Mindestanforderungen des Herstellers sind damit erfüllt. Die Software für unseren Test entspricht der Version 2.3.1+31549 mit einer Firmware Revision am Sensor von: 1.7.0. Diese Software entspricht der aktuellsten Version zum Zeitpunkt der Tests.

Ausgeliefert wird die Software mit einem Control Panel für Einstellungen und einen Visualizer (Abbildung 4.2) der die gelieferten Daten des Sensors visualisiert.

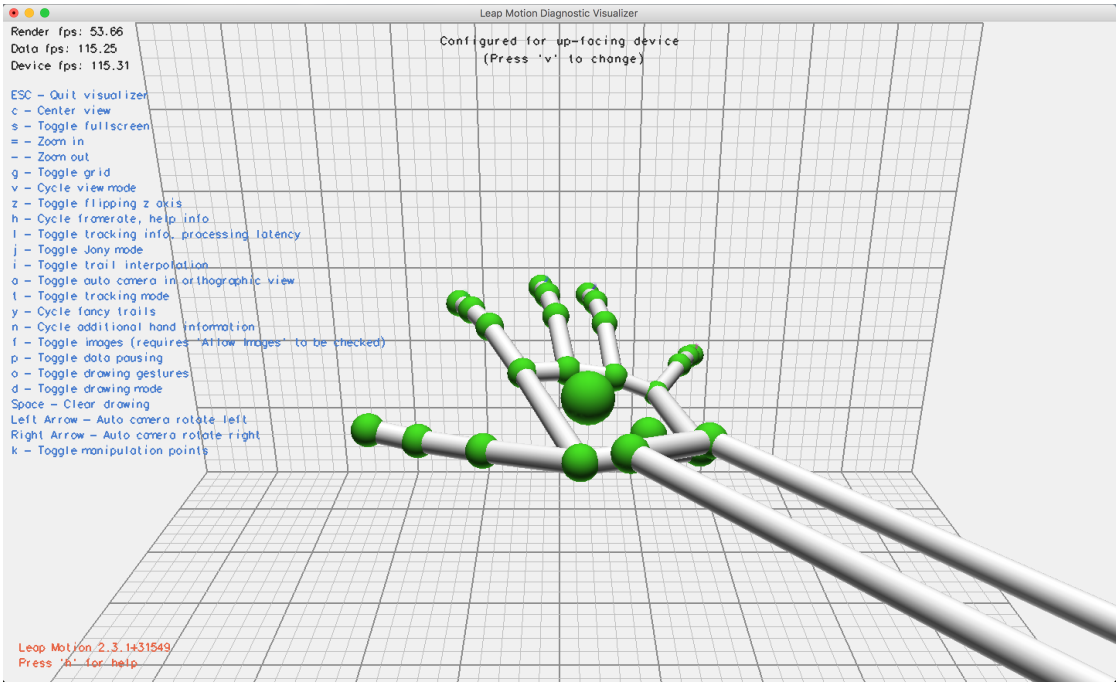


Abbildung 4.2: Leap Motion Sensor - Mit der Software ausgelieferter Diagnostic Visualizer zum Visualisieren der Sensor Daten.

Nach der erfolgreichen Installation und dem Testen des Sensors mit dem Visualizer steht der Recherche von Möglichkeiten für die Demo Applikation nichts mehr im Wege.

### Recherche

Der Applikation Store bietet bereits zahlreiche Software für den Bewegungssensor an. Ziel der Recherchearbeit ist es, bereits durch andere Entwickler zur Verfügung gestellte Applikationen für unsere Anforderungen zu testen, um den Aufwand für die Evaluierung zu reduzieren. Dadurch kann eine Mehrzahl an Technologien im Rahmen dieser Diplomarbeit getestet werden.

Eine Software des App Stores namens "Touch Everything"[Lead] erlaubt es jede Oberfläche in einen Touchscreen umzuwandeln. Ein Video kann unter `https://vimeo.com/111624691` angesehen werden. Diese Applikation würde die Anforderungen unserer Applikation erfüllen. Die Bereiche für die einzelnen Touches könnten den Segmenten auf einer haptischen Karte entsprechen um diese für die Genauigkeit beziehungsweise Empfindlichkeit zu evaluieren. Da diese Software zum aktuellen Zeitpunkt nicht für Mac OS X verfügbar war, konnte sie nicht im Rahmen der Diplomarbeit verwendet werden.

Der App Store bietet zahlreiche weitere Applikationen wie zum Beispiel "Better Touch Tool"[And] und "Touchless for Mac"[Leac] zur Steuerung des Computers. Diese erkennen sowohl Gesten, als auch Touch-Punkte. Auf Grund der Konfigurationsmöglichkeiten und der nicht Offenlegung des Source Codes kamen diese ebenfalls nicht in Frage.

Im Zuge der Recherche konnten aber auch Quellcodeteile gefunden werden, die für die eigene Entwicklung einer eigenen Demo App verwendet werden können. Diese sind lediglich auf der Leap Motion Website über das Entwickler-Mitgliedskonto erhältlich. So konnte für die Entwicklung der Applikation auf Codeteile für die Visualisierung der Hände zurückgegriffen werden.

### Verbindungsaufbau

Für die Implementierung wurde wegen der vorhandenen Code Segmente auf die Programmiersprache JavaScript gesetzt. Diese bietet die Möglichkeit rasch und unkompliziert Ergebnisse zu erzielen.

Um die SDK[2] für den Sensor in der gewünschten Programmiersprache herunterzuladen wird ein Entwickler-Konto vorausgesetzt. Dieses konnte kostenlos binnen weniger Minuten auf der Webseite angelegt werden.

Für die Entwicklung wurde als SDK LeapJS in der Version 0.6.4 verwendet, die zusammen mit HTML als Basis für unsere Implementierung dient.

Im ersten Schritt gilt es eine Verbindung mit dem Sensor herzustellen und die Daten, welche zur Verfügung gestellt werden auszuwerten. Der Sensor ruft bei jedem Schleifendurchgang die Funktion Loop auf, welche ihm den aktuellsten Frame zurück liefert und

[2] Software Development Kit

gibt diesen am Bildschirm aus. Der Code ist im Listing 4.1 ersichtlich. Durch die Verwendung der JavaScript Bibliothek Leap.js und der Funktion Loop können die Daten mit einer Geschwindigkeit von bis zu 60 Frames pro Sekunde verarbeitet werden [HCW14].

```
// Verbindung mit dem Controller herstellen
var controller = new Leap.Controller({enableGestures: true});

// Schleife zur Abarbeitung der Daten im Objekt Frame
controller.loop(function(frame) {
  // Ausgabe der Frame Daten am Bildschirm
  document.getElementById('out').innerHTML = "<div>"+ frame.
   dump() + "</div>";
}
```

Listing 4.1: Leap Demo App: Verbindungsaufbau zum Sensor und Ausgabe der Messdaten

Das Frame-Objekt bietet eine große Anzahl an Informationen, die in der Applikation genützt werden können. Diese sind in unterschiedliche Kategorien unterteilt. Die im Listing 4.1 aufgerufene Methode `frame.dump()` liefert nur einen Teil aller Informationen. Diese sind für die Demo App vorerst aber ausreichend. Abbildung 4.3 gibt eine Übersicht, welche Punkte der Hände und Finger als Daten zur Verfügung stehen. Eine komplette Übersicht aller Informationen des Frame Objektes kann der Online Dokumentation[3] der API entnommen werden.

- **Frame:** Dieser Block beinhaltet die Informationen, um welchen Frame es sich handelt. Jeder Frame verfügt über eine eindeutige ID und einen Zeitstempel. Zusätzlich wird die Anzahl von Händen, zeigenden Objekten und Gesten mitgeliefert.
- **Hände:** Ist ein Array mit allen erkannten Händen. Jeder Eintrag besteht aus einer eindeutigen ID, sowie der Information, ob es sich um eine linke oder rechte Hand handelt. Zusätzlich werden noch Informationen über die Position des Mittelpunkts der Handoberfläche und dessen Veränderung zum vorigen Frame angegeben.
- **Zeiger:** Gibt eine Liste von Zeigern (Finger oder Gegenstände) an, die in diesem Frame erkannt wurden. Pro Zeiger wird ebenfalls eine ID zur Referenzierung angegeben. Die Richtung, in welche gezeigt wird, wird durch einen Richtungsvektor angegeben. Dieser zeigt in die gleiche Richtung wie die Spitze des Zeigers. Zusätzlich wird noch die berechnete Breite des Zeigers angegeben. In der API wird der Zeiger als `Pointable` bezeichnet.
- **Gesten:** Ist eine Liste von allen erkannten Gesten des aktuellen Frames. Es gibt vier Gesten, die von der SDK erkannt werden: Kreise, Wischen, ScreenTap und KeyTap. Bei jeder Geste werden die IDs der verwendeten Hände und Zeiger angegeben, sowie dem Zustand der Geste.

[3] https://developer.leapmotion.com/documentation/javascript/api/Leap.Frame.html#Frame.gestures[]

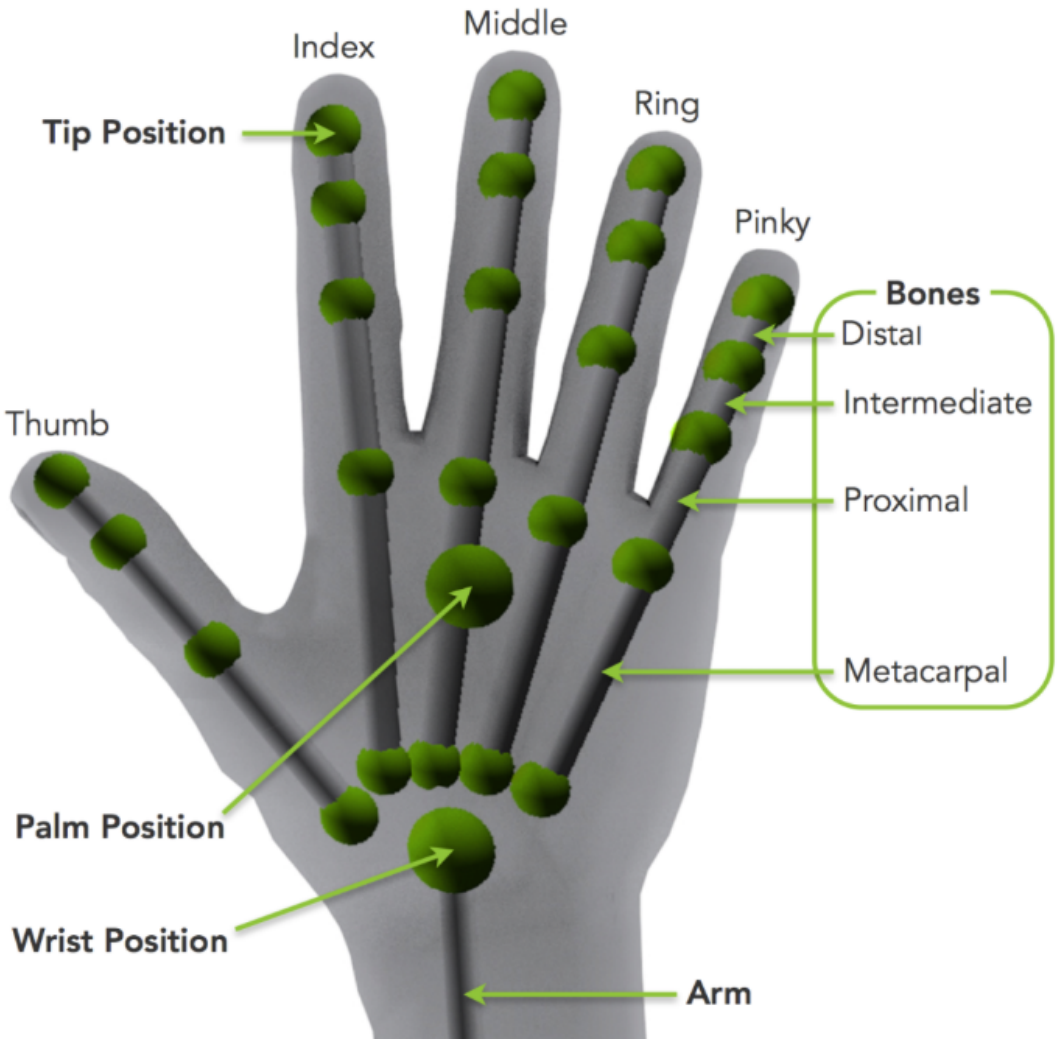


Abbildung 4.3: Leap Motion Sensor - Diese Abbildung zeigt alle Punkte einer Hand, die durch den Leap Motion Sensor gemessen werden können[Leab]. Diese Punkte dienen als Berechnungsgrundlage weiterer Informationen.

Im Listing 4.2 sieht man ein Beispiel einer Ausgabe eines Frame Objektes.

```
Frame Info:
Frame [ id:514938 | timestamp:430293322437 | Hand count:(1) |
    Pointable count:(5) | Gesture count:(1) ]

Hands:
  Hand (right) [ id: 33 | palm velocity
    :23.1569,20.7502,0.386159 | sphere center
```

```
    :-26.2538,161.831,73.372 ]

Pointables:
  Finger [ id:330 51.3156mmx | width:19.9388mm | direction
    :0.0235066,-0.341161,-0.939711 ]
  Finger [ id:331 57.9039mmx | width:19.0456mm | direction
    :-0.06737,-0.134929,-0.988562 ]
  Finger [ id:332 65.9768mmx | width:18.7053mm | direction
    :-0.26289,-0.519818,0.812821 ]
  Finger [ id:333 63.4385mmx | width:17.7993mm | direction
    :-0.404489,-0.445925,0.798461 ]
  Finger [ id:334 49.7346mmx | width:15.8107mm | direction
    :-0.578614,-0.364815,0.729463 ]

Gestures:
  CircleGesture [{"center":[-36.0461,204.372,-1.62885],"normal
    ":[-0.141597,0.131639,-0.981133],"progress":1,"radius
    ":8.90007,"id":42,"handIds":[33],"pointableIds":[331],"
    duration":0,"state":"start","type":"circle"}]
```

Listing 4.2: Leap Demo App: Beispielausgabe eines Frame Objektes

**Konfiguration der Touch-Fläche**

Durch die erfolgreiche Ausgabe der Informationen am Bildschirm ist der Grundstein für die Test Applikation gelegt. Im nächsten Schritt muss die Applikation die für den Test relevanten Daten bereitstellen. Im ersten Schritt wird dafür eine zweidimensionale Karte, welche als Touch Fläche dienen soll, definiert. Um die Applikation so flexibel wie möglich zu gestalten, werden beim Start der Software die drei Eckpunkte im 3-Dimensionalen Raum durch den/die BenutzerIn festgelegt. Der vierte Eckpunkt wird nach erfolgreicher Eingabe der ersten drei Eckpunkte automatisch berechnet. Diese vier Punkte ergeben Eckpunkte für die Touch Fläche. Durch einen Neustart der Applikation können die Eckpunkte neu kalibriert werden.

Das Festlegen der Punkte soll durch das Drücken der Tasten `0`, `1` und `2` auf der Tastatur erfolgen, während der Zeigefinger die jeweilige Ecke der Karte berührt. Der Einfachheit wegen wird davon ausgegangen, dass bei der Kalibrierung der Karte nur eine Hand über dem Sensor platziert ist. Bei mehreren Händen werden die Daten der ersten Hand für die Kalibrierung verwendet. Das Auslesen des Touch-Punktes erfolgt durch die Variable `TouchPosition` des Objektes `Pointable`. Die Kalibrierung der drei Punkte kann mehrfach wiederholt werden, durch erneutes Drücken der jeweiligen Ziffer.

```
if(frame.hands.length > 0)
{
  var hand = frame.hands[0];      // Erste Hand
```

```
  var finger = hand.fingers[1]; // Zeigefinger

  document.getElementById('point0').innerHTML = "<b>Point0:</b
   >" + finger.tipPosition;
}
```

Listing 4.3: Leap Demo App: Ausschnitt des Sourcecodes der für das Auslesen der Touchpunkte des Zeigefingers zuständig ist.

Sind die drei Punkte gesetzt, wird der vierte Punkt durch mathematische Methoden berechnet. Diese basieren auf einfachem Addieren und Subtrahieren der einzelnen Koordinaten. Mit diesen vier Punkten ist die Touch-Oberfläche (im weiteren auch Plane genannt) definiert.

#### Erkennung von Touches auf der Oberfläche

Für die Erkennung eines Touches auf dieser Oberfläche wird eine Funktion `distanceToPlane` definiert, die für die Berechnung der Entfernung zwischen dem aktuellen Touch-Punkt des Zeigefingers und der Plane zuständig ist. Ist diese Zahl positiv, befindet sich der Touch-Punkt des Zeigefingers vor der konfigurierten Fläche. Bei einem negativen Wert befindet sich dieser hinter der Touch-Fläche.

Ein Touch wird gewertet, wenn die Distanz zur Plane unter einen Schwellwert von `2` fällt. Dieser Schwellwert ist wichtig, da durch unterschiedlich starkes Aufdrücken des Fingers die Distanz zur Karte leicht variieren kann. Der hier angeführte Wert hat sich für die Tests als sehr zuverlässig erwiesen.

#### Ausgabe des Touch-Segmentes

Für die bevorstehenden Tests wird eine Genauigkeit von mindestens 1cm$^2$ zum Testen benötigt. Zu diesem Zweck wird die berechnete Touch Oberfläche in 29x20 Segmente geteilt.

Wird ein Touch erkannt, wird die X- und Y- Position auf der Oberfläche berechnet und diese einem Segment zugeteilt. Die Segmente auf der X-Achse werden mit Buchstaben von `A` bis `AC`, und auf der Y-Achse von `1` bis `20` beschriftet

Ein Beispiel der ausgegeben Informationen ist im Listing 4.4 gelistet.

```
finger tip: 0 x:-33 y:146 z:24     // Daumen
finger tip: 1 x:-11 y:197 z:-28    // Zeigefinger
finger tip: 2 x:-23 y:142 z:63     // Mittelfinger
finger tip: 3 x:-9 y:138 z:72      // Ringfinger
finger tip: 4 x:3 y:144 z:69       // Kleiner Finger

CalibPoint #1 x:-91 y:270 z:-53    // Eckpunkt Links-Oben
CalibPoint #2 x:141 y:256 z:-47    // Eckpunkt Rechts-Oben
```

```
CalibPoint #3 x:-101 y:138 z:-7   // Eckpunkt Links-Unten
CalibPoint #4 (calculated) x:131 y:124 z:-1 //Eckpunkt Rechts-
   Unten

Distance to touchplane: 0.80337

Touch Position: x:80 y:73, Segment: J - 12
```

Listing 4.4: Leap Demo App: Beispielausgabe der berechneten Punkte der Touchplane.

### Visualisierung der Daten

Um die Richtigkeit der gemessenen Punkte zu prüfen, werden die Daten mit Hilfe der JavaScript Bibliothek `Three.js`[4] visualisiert. Three.js ist eine 3D Bibliothek, die es auch für Neueinsteiger ermöglicht, relativ schnell Ergebnisse zu liefern.

Für die Kombination des Leap Motion Sensors und Three.js gibt es eine Standardvorlage (im weiteren Verlauf als Boilerplate bezeichnet), die für JavaScript vorhanden ist. Die Software[5] verwendet Three.js um eine Hand und deren Finger in 3D anzuzeigen. Die Daten werden in Echtzeit verarbeitet und visualisiert. Diese Boilerplate, programmiert von Jaanga Software, steht unter der MIT Lizenz[6] zur Verfügung.

Das Zusammenführen dieser Boilerplate mit der bereits entwickelten Demo App ermöglicht die Chance eine visualisierte Interpretation der Daten zu erlangen.

Für die Verwendung von Three.js wird zuerst ein Web-Renderer mit der gewünschten Größe definiert und in den HTML-Sourcecodes eingehängt. Für die Ansicht der gerenderten Elemente wird noch eine Kamera und deren Ausrichtung definiert und eine Szene erstellt. Alle darzustellenden Elemente werden dieser Szene hinzugefügt und durch den Renderer dargestellt. Zur besseren Orientierung wird ein Raster und die Koordinatenachsen zur Szene hinzugefügt. Ein Ausschnitt des Sourcecodes für die Konfiguration des Webrenderers ist im Listing 4.5 angeführt.

```
// Fenster zur Darstellung
    renderer = new THREE.WebGLRenderer( { alpha: 1, antialias:
   true, clearColor: 0xffffff } );
    renderer.setSize( window.innerWidth, window.innerHeight );
    document.body.appendChild( renderer.domElement );

// Kamera Position
    camera = new THREE.PerspectiveCamera( 45, window.innerWidth
    / window.innerHeight, 1, 5000 );
```

[4] http://threejs.org

[5] https://developer.leapmotion.com/gallery/boilerplate-for-three-js-and-leapjs

[6] http://jaanga.github.io/home/r4/index.html#http://jaanga.github.io/jaanga-copyright-and-mit-license.md

```
    camera.position.set( 000, 400, 600 );
    controls = new THREE.TrackballControls( camera, renderer.
   domElement );

// Szene Erstellung
    scene = new THREE.Scene();

// Raster in der Szene einbinden
    var geometry = new THREE.BoxGeometry( 500, 2, 500 );
    material = new THREE.MeshNormalMaterial();
    var mesh = new THREE.Mesh( geometry, material );
    mesh.position.set( 0, -1, 0 );
    scene.add( mesh );

    mesh = new THREE.GridHelper( 250, 10 );
    scene.add( mesh );

// Achsen anzeigen
    var axis = new THREE.AxisHelper( 250 );
    scene.add( axis );

// Szene mit der definierten Kameraeinstellung rendern
    renderer.render( scene, camera );
```

Listing 4.5: Leap Demo App: Konfiguration des Three.js Renderer

Die Visualisierung der Hand funktioniert direkt von der Boilerplate heraus. Die Software muss nun lediglich um Funktionen für die Visualisierung der Kalibrierpunkte und der Touch-Fläche erweitert werden. Die Kalibrierpunkte werden durch einfache Quader dargestellt, während die Touch-Fläche aus zwei Dreiecken zusammengesetzt wird.

Nach der Visualisierung und Prüfung der Software befindet sich diese in einem Zustand, in dem Tests für die Validierung durchgeführt werden können. Die vollständige Software ist auf Github[7] ersichtlich. Abbildung 4.4 zeigt die fertig aussehende Demo Applikation.

### 4.2.3 Evaluierung

Für die Evaluierung werden mit Hilfe der zuvor entwickelten Software eine Reihe an Versuchen durchgeführt. Standardmäßig ist der Leap Motion Sensor dafür konzeptioniert, mit dem Sensor nach oben auf dem Tisch zu stehen. In dieser Position ist die Software in der Lage die darüber platzierten Hände und Finger zu erkennen. Unter diesen Bedingungen ist es nur möglich mit einer Touch-Karte zu agieren, wenn diese senkrecht vor dem Sensor angebracht ist. Nichts desto trotz muss hier ebenfalls die Genauigkeit geprüft werden.

[7] https://github.com/derro/LeapTouchScreen

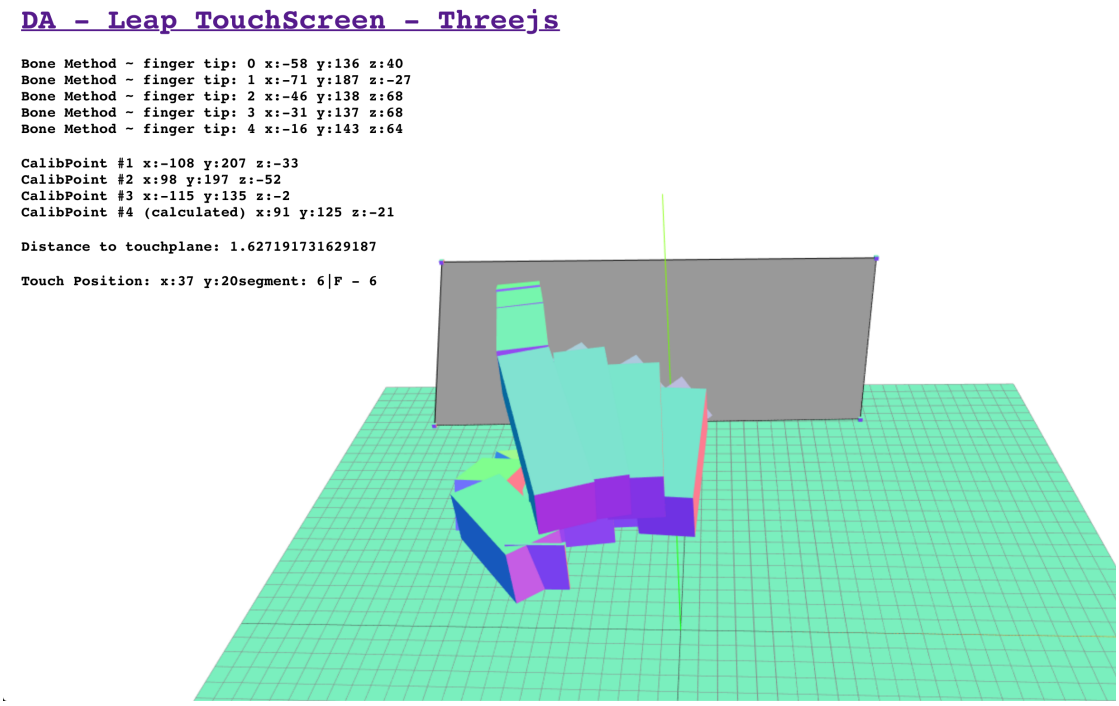


Abbildung 4.4: Leap Motion Sensor - Finale Darstellung der Demo Applikation inklusive einer definierten Touch-Fläche und Ausgabe des Touch-Segmentes.

| Versuch | Position des Sensors | Neigung | Position der Karte |
|---|---|---|---|
| Nr. 1 | Nach oben zeigend | horizontal, 0°Grad | Senkrecht vor dem Sensor |
| Nr. 2 | Nach unten zeigend | horizontal, 0°Grad | Waagrecht am Tisch liegend |
| Nr. 3 | Nach unten zeigend | seitlich, 120°Grad vorne | Waagrecht am Tisch liegend |
| Nr. 4 | Nach unten zeigend | seitlich, 90°Grad vorne | Waagrecht am Tisch liegend |

Tabelle 4.1: Übersicht über die Experimente zur Evaluierung des Leap Motion Sensors.

Zusätzlich werden Versuche durchgeführt, in denen der Multi-Touch Sensor in anderen Positionen gebracht wird.

Die Tabelle 4.1 zeigt eine Übersicht über die einzelnen Experimente und deren Unterscheidungsmerkmale.

### Versuchsaufbau

Um bei den unterschiedlichen Versuchen vergleichbare Ergebnisse zu bekommen, werden gewisse Parameter für alle Experimente festgelegt. Die Karte, die für die Erkennung verwendet wird, ist ein weißes Blatt A4 mit einem Raster im 1cm Abstand. Der Raster

| Nummer | Soll X | Soll Y | Kommentar |
|---|---|---|---|
| Nr. 1 | 1 | 1 | Eckpunkt links oben |
| Nr. 2 | 29 | 1 | Eckpunkt rechts oben |
| Nr. 3 | 1 | 20 | Eckpunkt links unten |
| Nr. 4 | 29 | 20 | Eckpunkt rechts unten |
| Nr. 5 | 15 | 1 | Mittelpunkt (Mitte, Oben) |
| Nr. 6 | 15 | 20 | Mittelpunkt (Mitte, Unten) |
| Nr. 7 | 15 | 10 | Mittelpunkt (Mitte, Mitte) |
| Nr. 8 | 2 | 1 | Zufallspunkt 1 |
| Nr. 9 | 22 | 5 | Zufallspunkt 2 |
| Nr. 10 | 27 | 8 | Zufallspunkt 3 |
| Nr. 11 | 6 | 5 | Zufallspunkt 4 |
| Nr. 12 | 11 | 2 | Zufallspunkt 5 |
| Nr. 13 | 19 | 15 | Zufallspunkt 6 |
| Nr. 14 | 7 | 18 | Zufallspunkt 7 |
| Nr. 15 | 29 | 20 | Zufallspunkt 8 |
| Nr. 16 | 9 | 17 | Zufallspunkt 9 |
| Nr. 17 | 19 | 19 | Zufallspunkt 10 |

Tabelle 4.2: Leap Motion Demo App: Übersicht der zu messenden Punkte bei den Experimenten.

ist auf der X-Achse von 1 bis 29 und auf der Y-Achse von 1 bis 20 nummeriert. In jedem Experiment werden 17 Rastersegmente getestet. Diese 17 Punkte bestehen aus den vier Eckpunkten, sowie der drei Punkten in der Mitte und zehn weiteren Zufallswerten.

Die Tabelle 4.2 gibt eine Übersicht über die zu testenden Punkte. Grafik 4.5 zeigt eine graphische Darstellung.

**Versuch Nr. 1**

Für dieses Experiment ist der Leap Sensor waagrecht mit der Blickrichtung nach oben auf dem Tisch ausgerichtet und 6.5 cm von der Wand entfernt, auf der die Karte angebracht ist. Die Karte selbst befindet sich in einer Höhe von 15.5 cm über dem Sensor an der Wand. Der Versuchsaufbau ist in der Grafik 4.6 dargestellt.

Von den 17 Vermessungspunkten konnten 16 Stück fehlerfrei erkannt werden. Bei einem Punkt (Nr. 4) ist eine Abweichung von +1 auf der X-Achse gemessen worden. Eine graphische Übersicht der Messergebnisse sind der Abbildung 4.6 zu entnehmen. In Summe ist das Ergebnis sehr zufriedenstellend. Als Nachteil erweist sich die Handhabung mit der Karte. Die Interaktion mit dieser Touch Fläche ist auf Grund der Höhe und Montage an der Wand sehr eingeschränkt möglich.

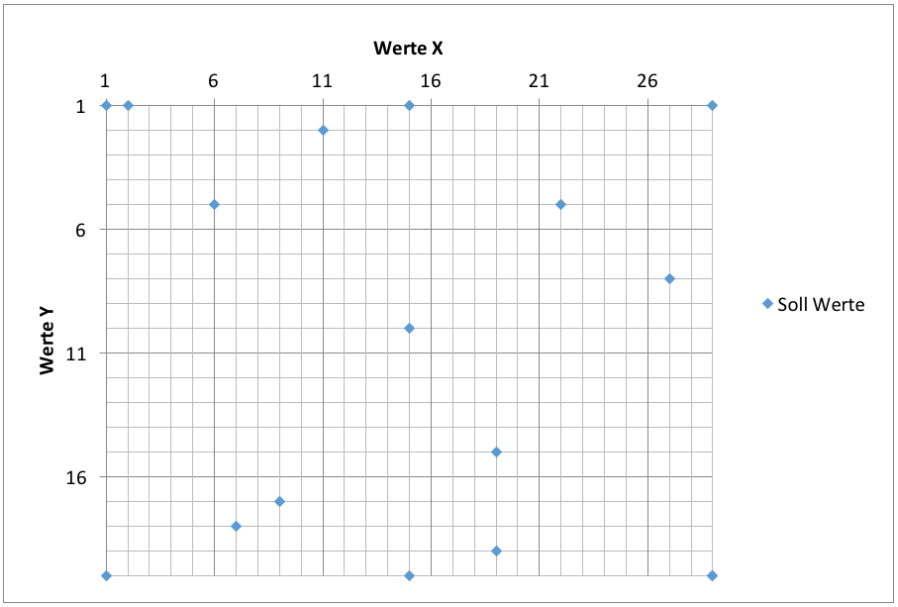


Abbildung 4.5: Graphische Darstellung der gemessenen Punkte bei den Experimenten.

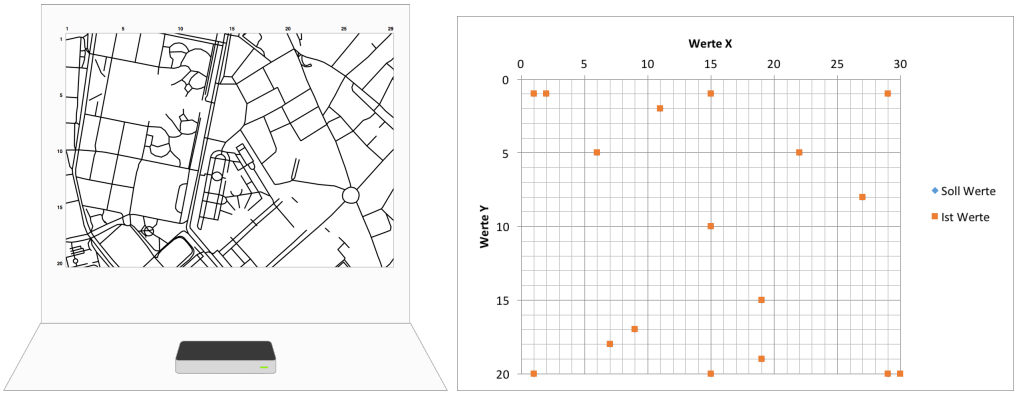


Abbildung 4.6: Versuchsaufbau Experiment Nr. 1 (links) mit den Ergebnissen der Messungen (rechts).

### Versuch Nr. 2

Beim Experiment Nummer 2 wurde der Sensor kopfüber an einer Vorrichtung befestigt und mittig über den am Tisch liegenden Blatt Papier ausgerichtet. Der Sensor ist dabei 23.5 cm über dem Tisch ausgerichtet.

Auf Grund der starken Reflexion des Tisches und der Karte des ausgesendeten Infrarot-Lichtes des Sensors, ist der Kontrast des erkannten Bildes sehr niedrig. Durch diesen

niedrigen Kontrast ist eine Erkennung der Hände und Finger im Bereich von 5 Zentimeter über der Karte nicht möglich. Zur Lösung dieses Problems wurde der Tisch mit einem lichtabsorbierenden Stoff (schwarzer Filz) ausgelegt und die Karte ebenfalls aus Filz hergestellt. Der Versuchsaufbau wird in der Abbildung 4.7 dargestellt.

Von den 17 Vermessungspunkten konnten 10 Stück fehlerfrei erkannt werden. Bei 6 Stück waren die Y-Werte um $-1$ verschoben und bei einem Stück der X-Wert um $+1$. Auf Grund der hohen Zahl an Abweichungen der Y-Werte wurde der Sensor um 5 weitere Zentimeter angehoben. Die Ausrichtung wurde anstelle von mittig über dem Blatt auf 1/3 über der südlichen Kante ausgerichtet. Ebenfalls wurde die Touchfläche neu kalibriert.

Im zweiten Messschritt, unter den oben genannten Voraussetzungen, konnte das Ergebnis deutlich verbessert werden. Es traten 15 positive Ergebnisse und lediglich 2 negative Messungen auf. Die Übersicht über die Messergebnisse sind der Abbildung 4.8 zu entnehmen.

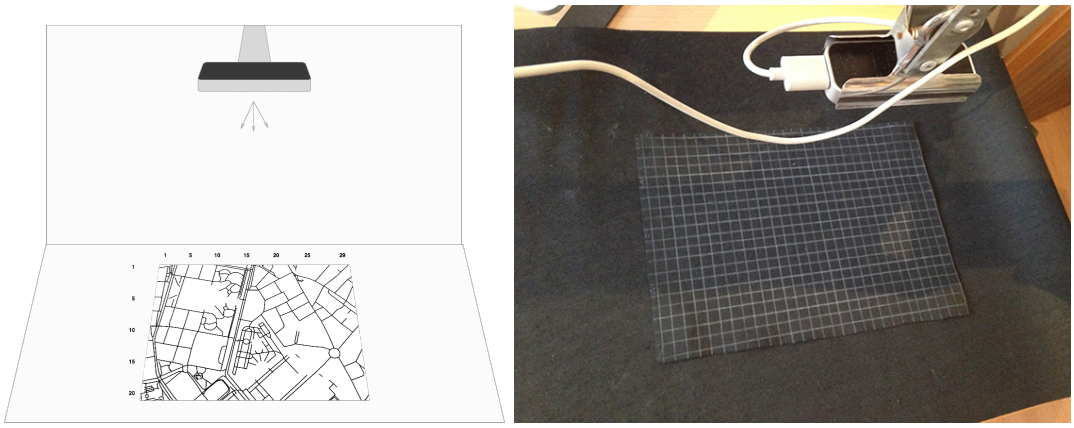

Abbildung 4.7: Schematischer Versuchsaufbau Experiment Nr. 2 (links). Versuchsaufbau in der Praxis mit einer Filzunterlage (rechts).

**Versuch Nr. 3**

In der dritten Variante wurde der Sensor um 110°Grad geneigt und zehn Zentimeter über der Karte befestigt. In dieser Position wurde der Sensor fünf Zentimeter vor der Karte angebracht.

Die Messung lieferte 10 erfolgreiche und 7 falsche Punkte. Im Vergleich zu den anderen Experimenten traten hier erstmals sowohl Abweichungen auf der X- als auch Y-Achse bei einigen Punkten auf. Vor allem die Werte auf der rechten Seite der Karte wurden falsch erkannt. Die Abbildung 4.9 zeigt den Versuchsaufbau und die Messergebnisse.

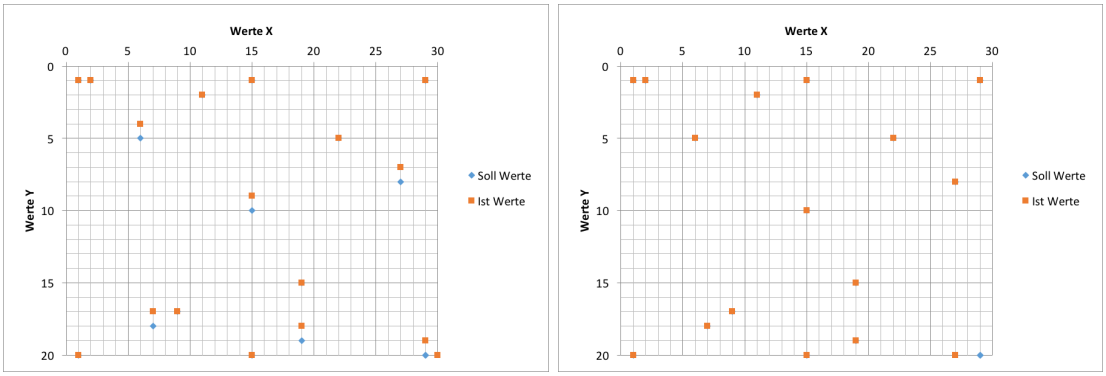


Abbildung 4.8: Messergebnisse der ersten Vermessung auf einer Höhe des Sensors von 23.5 cm (links) und der zweiten Vermessung auf einer Höhe von 28.5 cm (rechts).

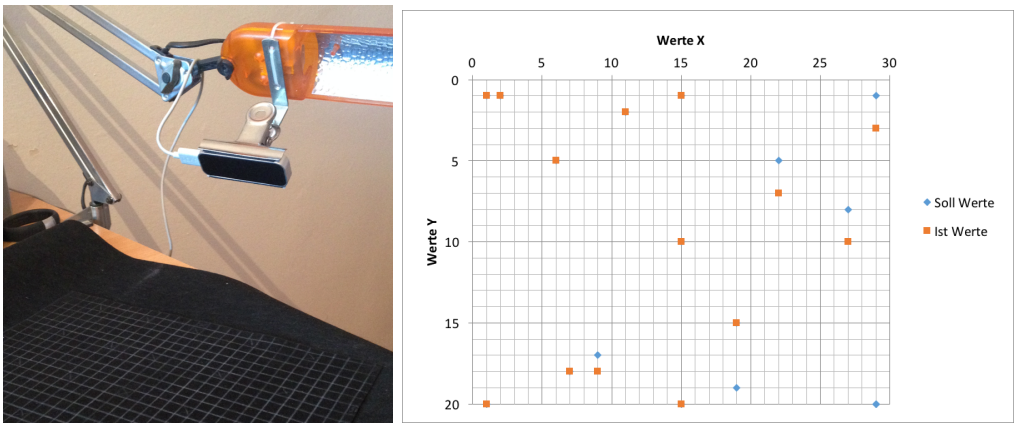


Abbildung 4.9: Versuchsaufbau Nr. 3 in der Praxis (links). Die Messergebnisse des Experimentes (rechts).

### Versuch Nr. 4

Ähnlich dem Versuch 3 wurde auch bei Versuch 4 der Sensor schief geneigt vor der Karte angebracht. Der Winkel entspricht bei diesem Versuch 90°. Der Sensor wurde in einer Höhe von 8 cm und 10 cm vor dem Raster ausgerichtet.

Die Ergebnise der Messung waren 13 richtig erkannte Punkte und 4 Fehlinterpretationen. Diese sind in der Grafik 4.10 zusammengefasst.

## 4.2.4 Fazit

Nach der Entwicklung der Demo Applikation und der Durchführung der Experimente ist es möglich alle Anforderungen zu prüfen und den Sensor zu bewerten.

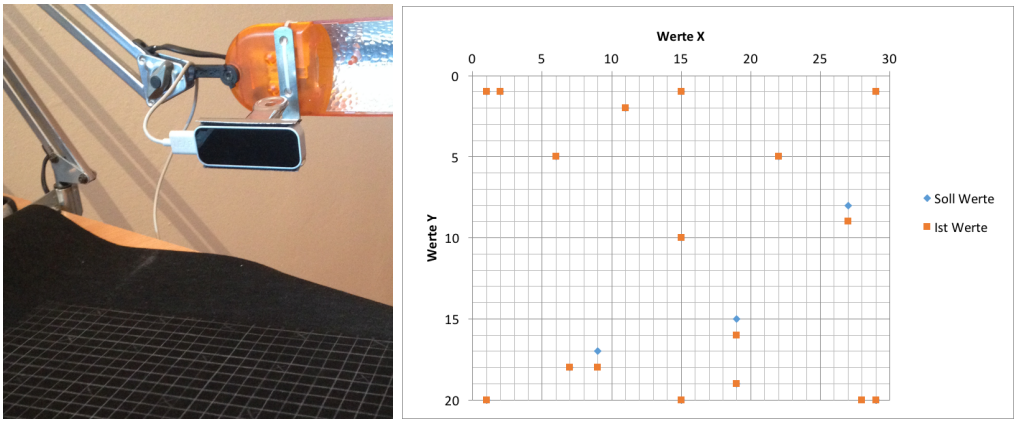


Abbildung 4.10: Versuchsaufbau Nr. 4 in der Praxis (links). Die Messergebnisse des Experimentes (rechts).

- **Größe des Sensors:** Der Interaktionsraum des Sensors stellt mit 60cm x 60cm x 60cm genug Platz dar, für die Interaktion mit einer Karte in einer Größe von mindestens A4. Bei allen Experimenten war der Sensor so angebracht, dass eine Interaktion mit der Karte problemlos, wenn auch nicht immer natürlich, möglich war.

- **Empfindlichkeit:** Da der Sensor auf die Reflexion der LEDs für die Erkennung der Hände angewiesen ist, kann der Sensor entweder oberhalb der Karte (Experimente 2-4) oder vor dem Sensor (Experiment 1) angebracht werden. Für die Variante oberhalb, muss der Tisch und die Karte eine nicht reflektierende Fläche aufweisen. Dies kann bei der Herstellung von haptischen Karten zu Schwierigkeiten führen.

- **Genauigkeit:** Die Experimente mit der höchsten Genauigkeit waren Experiment 1 mit 94.12% und Experiment 3 mit 88,24% Wahrscheinlichkeit. Bei einer Entwicklung eines Prototypen mit dieser Technologie, sollte man sich für die Parameter und Versuchsaufbauten dieser beiden Experimente entscheiden. Eine Übersicht über die Ergebnisse der Experimente wird in der Abbildung 4.12 veranschaulicht.

- **Multi-Touch Unterstützung:** Bei den Tests konnten beide Hände, als auch alle 10 Finger erkannt werden.

- **Gesten Unterstützung:** Die API unterstützt vier Gesten: Kreis- und Wischbewegungen sowie Tippen nach vorne und unten. Weitere Gesten müssen durch eigene Codezeilen implementiert werden.

- **Physischer Aufbau:** Durch die Kalibrierung mit der Karte ist eine Ausrichtung des Sensors möglich. Die Kalibrierung muss nach jedem Aufbau erneut durchgeführt werden.

- **Wartung:** Die Wartung des Sensors erfolgt durch das Reinigen des Glases über der Kamera und der LEDs. Eine Kalibrierung mit dem Betriebssystem ist nur bei anhaltenden Missinterpretationen notwendig. Eine Kalibrierung der Ausrichtung zur Karte ist in den meisten Fällen ausreichend.
- **API:** Es werden APIs in den Sprachen JavaScript, Unity, C#, C++, Java, Phyton, Objective-C und Unreal angeboten. Eine Implementierung mit JavaScript stellte sich durch die Demo Applikation als ausreichend heraus.

Alles zusammen ergibt der Leap Motion Sensor ein sehr kompakte Tool zur Erkennung von Händen, Finger und Gesten. Da bei den Experimenten keine 100%igen Erkennungsrate erreicht werden konnte, kann man bei der Verwendung dieser Technologie für die Erprobung des Konzepts mit falschen Interpretationen der Punkte beziehungsweise Gesten rechnen. Des Weiteren ist die Anzahl, der von Haus aus unterstützten Gesten, sehr gering. Für den Aufbau muss man spezielle Gerüste zur Verfügung stellen, um den Sensor immer ideal zur Karte auszurichten. Bei der Betrachtung des Architekturkonzepts von Echtler et al sieht man, dass man bei der Software für den Leap Motion Sensor in der Interpreationsschicht ansetzten muss um alle benötigten Gesten zu unterstützen[EK08].

Eine Implementierung des Prototypen mit dieser Technologie ist möglich, aber für das zu Grunde liegende Konzept zu ungenau. Es wird zur Evaluierung weiterer Technologien geraten.

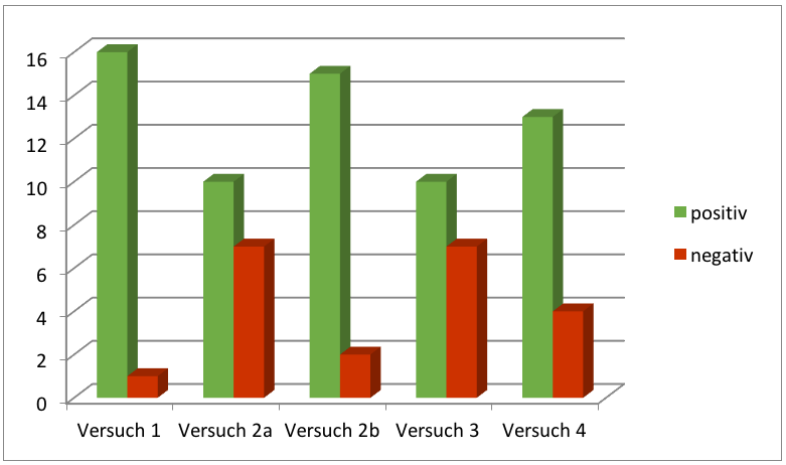


Abbildung 4.11: Übersicht über die Anzahl der richtigen und fehlerhaften Messungen bei den Experimenten.

## 4.3 iPad Pro

Das Apple iPad Pro ist ein Tablet der Firma Apple, das im Herbst 2015 als Nachfolger der iPad-Tablet Serie veröffentlicht wurde. Dieses Tablet ist im Vergleich zu seinen

Vorgängern mit einem 12,9"Retina Display ausgestattet. Das entspricht einer Sensorgröße von 26,2 x 19,7cm, was ungefähr mit der Maße eines A4-Blattes vergleichbar ist. Im Vergleich dazu, bietet das iPad Air 2 ein 9,7"bzw. das iPad Mini 4 ein 7,9"Multi-Touch Display. Das iPad Pro ist einer Auflösung von 5,6 Millionen Pixeln das hochauflösendste Display der iPad Serie. Das Tablet ist mit einem A9X-Chip mit 64-Bit Architektur und einem M9 Motion Co-Prozessor ausgestattet. Zusätzlich ist das iPad Pro mit einer Kamera auf der Vorder- und Rückseite ausgestattet.

Durch die neue Größe des iPad Pro bietet es eine ideale Grundlage für die Analyse des Sensors für das Konzept von Sommersguter.

Im Frühling 2016 wurde von Apple ein zweites iPad Pro unter dem gleichen Namen mit einer Display-Größe von 9,7"vorgestellt. Alle in dieser Diplomarbeit genannten iPad Pro Versionen beziehen sich auf das iPad Pro mit einem 12,9"Display.

### 4.3.1 Vorgehen

Beim Vorgehen der Prüfung des iPad Pro wird ähnlich der Prüfung des Leap Motion Sensors vorgegangen.

- Im ersten Teil 4.3.2 wird eine Demo Applikation erstellt, welche die Touch-Informationen des Sensors entgegennimmt und verarbeitet. Das Ziel dieser Applikation ist es, im Endstadium Touches auf einer Karte zu erkennen und diese der richtigen Position auf der Karte zuzuordnen.
- Im Vergleich zum Leap Motion Sensor wird bei dieser Technologie die Karte auf den Sensor gelegt. Die Erkennung der Touch-Positionen sowie der Gesten muss hier trotz der dazwischenliegenden Karte erfolgen. Die Prüfung über die Genauigkeiten bei den einzelnen Positionen ist hierbei sehr wichtig. Diese Punkte werden im Kapitel 4.3.3 behandelt.
- Der letzte Schritt ist im Abschnitt 4.3.4 alle Anforderungen am Multi-Touch Sensor zu prüfen. Diese stellen die Basis für die Entscheidung dar, ob dieser Sensor für einen Implementation des Konzeptes geeignet ist.

### 4.3.2 Demo Applikation

#### Vorbereitung und Installation

Als Entwicklungsumgebung für unsere Demo Applikation wurde Xcode 7.1.1 auf einen MacBook Pro Retina (Mid 2012) gewählt. Die Programmiersprache für die Demo Applikation ist Swift. Die Installation von Xcode erfolgte via der Installation im Mac OS X App Store. Um das iPad Pro in der Entwicklungsumgebung als Testgerät verwenden können, muss dieses lediglich mit einem USB-Kabel an den Computer angeschlossen werden. Danach steht es im Programm Xcode als Testgerät zur Verfügung.

### Basisapplikation

Um sich mit den Möglichkeiten des iPad Pro vertraut zu machen, wird im ersten Schritt eine simple "One-Page Applikation" erstellt. Ziel der Applikation ist zu Beginn möglichst viele Informationen über die ausgeübten Touches am iPad Pro auszugeben.

Zu diesem Zweck wird auf der GUI die aufgerufene Methode für die Toucherkennung, die Anzahl der Berührungen, die Anzahl der Taps und die letzte Position ausgegeben. Um diese Informationen zu erlangen, werden die folgenden vier Funktionen für die Touch-Erkennung im Code überlagert[8]: `touchesBegan`, `touchesMoved`, `touchesEnded` und `touchesCancelled`.

- `touchesBegan` wird aufgerufen, wenn ein oder mehrere Finger in der Applikation den Sensor berühren. Diese Touches werden in der Applikation als `UITouch`-Objekt einem eindeutigen Event zugeordnet.
- `touchesMoved` wird bei der Bewegung der Finger, welche mit einem Event assoziiert sind, aufgerufen.
- `touchesEnded` gibt an, dass ein oder mehrere Finger die View verlassen haben. In diesem Fall endet das zugehörige Event durch das Beenden der Touches.
- `touchesCancelled` wird aufgerufen, wenn ein System Event ein Touch Event abbricht. Ein Beispiel dafür ist ein Face-Time Anruf, oder das Drücken des Home-Buttons.

Alle diese oben genannten Funktionen sind in der originalen Implementierung leer und haben keine Funktionalität. Eine Überlagerung dieser für den Prototypen ist daher ohne Auswirkungen möglich. Im Listing 4.6 ist die Überlagerung der Funktion `touchesBegan` angeführt. Die Überlagerungen der anderen drei Funktionen verhalten sich äquivalent dazu.

```
override func touchesBegan(touches: Set<UITouch>, withEvent
   event: UIEvent?) {
  // Ausgabe der Anzahl an Touches
  touchTotal.text = String((event?.allTouches()?.count)!)

  if let touch = touches.first {
    // Ausgabe der Anzahl an Taps
    let tapCount = touch.tapCount
    tapStatus.text = "\(tapCount) taps"

    // Auslesen der Position und Ausgabe deren
    let point = touch.locationInView(self.view)
```

[8] Ersetzen der Originalimplementierung der Funktion durch einen eigenen Code.

```
        singleTouchX.text = String(point.x)
        singleTouchY.text = String(point.y)
    }
}
```

Listing 4.6: iPad Pro Demo App: Überlagerung der Funktion touchesBegan

Dieser Code ermöglicht bereits die oben definierten Informationen auf der Applikation auszugeben.

Per Default-Wert sind Multi-Touches in der Software deaktiviert. Um in späterer Folge mehrere Touches bzw. Gesten zu erkennen, muss die Eigenschaft `multipleTouchEnabled` auf `true` gesetzt werden.

**Erkennung von Touches**

Das Konzept sieht vor, dass blinde Personen mit einer haptischen Karte interagieren und diese mit all ihren zehn Fingern erkunden können. Um sich mehr Informationen zu einem Punkt ausgeben zu lassen, werden alle Finger, bis auf jenen der auf den gewünschten Ort zeigt, von der Karte gehoben. Nach einer kurzen Pause erfolgt die akustische Ausgabe der Informationen zu diesem Punkt. Dieser Ablauf ist eines der Kernelemente des Konzeptes und soll daher bereits in der Demo Applikation in groben Zügen abgebildet sein.

iOS 9.1 stellt für die Erkennung von Gesten einen `UIGestureRecognizer` zur Verfügung. Diese abstrakte Klasse besitzt sieben unterschiedliche konkrete Gesten-Recognizer:

- `UITapGestureRecognizer` erkennt einzelne oder mehrfache Taps. Die Mindestanzahl der Taps und die Anzahl an benötigten Touches ist konfigurierbar. Werden diese Werte unterschritten, wird keine Geste erkannt.
- `UIPinchGestureRecognizer` erkennt das zu- oder auseinander Bewegen zweier Finger.
- `UIRotationGestureRecognizer` erkennt Drehbewegungen zweier gegenüberliegenden Punkte in die gleiche Richtung.
- `UISwipeGestureRecognizer` erkennt eine Wischgeste von einem oder mehreren Fingern in eine oder mehrere Richtungen. Die Mindestanzahl an Touches für eine positive Erkennung ist dabei konfigurierbar.
- `UIPanGestureRecognizer` erkennt eine Ziehbewegung eines oder mehrere Finger über eine Strecke. Die Mindest- und Maximalanzahl an Touches die für die Erkennung notwendig sind, ist konfigurierbar.
- `UIScreenEdgePanGestureRecognizer` erkennt eine Ziehbewegung welche in einer Ecke des Bildschirms beginnt. Die gültigen Ecken dafür sind konfigurierbar.

- `UILongPressGestureRecognizer` erkennt langanhaltende Touches. Die minimale Dauer und die Anzahl der benötigten Touches ist konfigurierbar, wie die Anzahl an benötigten Taps und die erlaubte Abweichung zum Startpunkt.

Einzig der `UILongPressGestureRecognizer` bieten in groben Zügen die notwendigen Eigenschaften, um die oben definierte Anforderung, zu erfüllen.

Das Testen der Implementierung dieses Reocgnizers zeigte, dass sich die Anforderung damit nicht abdecken lässt. Ein reines Tippen und Halten wird erfolgreich erkannt, aber wenn ein Finger auf der Karte bewegt wird und erst für die Informationsausgabe auf einem Punkt stehen bleibt, wird dies nicht erkannt. Lediglich wenn der Finger nach dem Bewegen kurz vom Touchdisplay gehoben und neu aufgesetzt wird.

Eine Lösung bietet eine individuelle Implementierung. Dazu wird auf die oben überlagerten Methoden `touchesBegan`, `touchesMoved`, `touchesEnded` und `touchesCancelled` zurückgegriffen. In der Methode `touchesBegan` wird ein Countdown (Objekt `NSTimer`) initialisiert und gestartet. Läuft der Countdown ab, wird eine Methode für die Ausgabe der Position aufgerufen. Beim Aufruf der Methode `touchesMoved` wird der Countdown zurück auf den Startwert gesetzt. Die Methoden `touchesEnded` und `touchesCancelled` deaktivieren den Countdown. Der Startwert für den Countdown wird in einer Variable mit dem Default-Wert von einer Sekunde hinterlegt.

#### Ausgabe des Touch-Segmentes

Wie bereits bei der LeapMotion Demo Applikation, wird auch für diese Applikation die Oberfläche in Segmente zur Genauigkeitsmessung unterteilt. In diesem Fall in einem Raster von 27x20. Die Abweichung von 2 Segmenten in der Breite gegenüber des Leap Motion Sensors, ergibt sich auf Grund der unterschiedlichen Sensorgröße. Beim iPad ist diese eindeutig durch das Display definiert, während beim Leap Motion Sensor das komplette A4-Blatt im Interaktionsraum zur Verfügung stand.

Wird ein Touch erkannt, wird die X- und Y-Position dessen für die Berechnung des Segmentes herangezogen.

#### Audio Ausgabe

Für die anstehenden Experimente wird das Display des iPad Pros mit der haptischen Karte verdeckt sein, um zu testen, ob auch Touches durch die Karte hinweg erkannt werden können. Ein Ablesen der Werte ist daher ohne das Entfernen der Karte nicht möglich. Aus diesem Grund wird die Position am iPad zusätzlich akustisch ausgegeben.

Die akustische Ausgabe erfolgt über die Klasse `AVSpeechSynthesizer` des Paketes `AVFoundation`. Diese Klasse beinhaltet alle notwendigen Parameter zur Konfiguration und Methoden zur Steuerung der Sprachausgabe. Die Konfiguration erfolgte durch die hinterlegten Default-Werte.

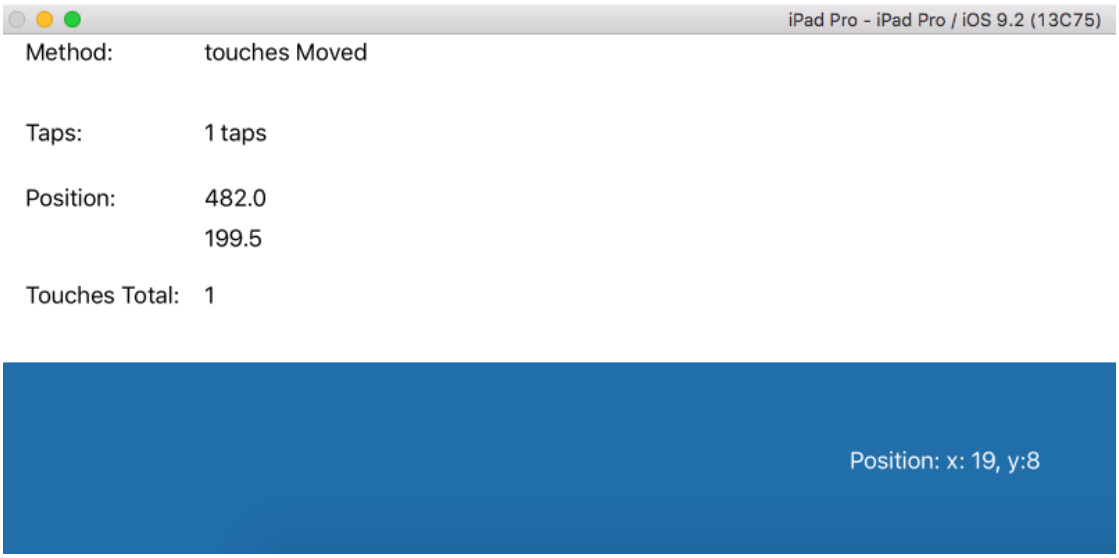


Abbildung 4.12: Ausgabe der Touch Informationen und der Position des POI.

### 4.3.3 Evaluierung

Dieses Kapitel beschäftigt sich mit der Evaluierung der Demo Applikation für das iPad Pro. Ziel ist es verschiedene Szenarien durchzuführen, um eine Vergleichsbasis zu den Versuchen des Leap Motion Sensors herzustellen.

#### Versuchsaufbau

Für den Versuch wird die gleiche Ausgangsbasis wie bereits bei dem Leap Motion Sensor gewählt. Für die Karte wird ein weißes A4-Blatt (80 Gramm) mit dem Bereich des Sensors gekennzeichnet. In diesem Bereich wird im Abstand von 1cm ein Raster aufgezeichnet. Die Segmente werden auf der X-Achse von 1 bis 26 und auf der Y-Achse von 1 bis 20 nummeriert. Zusätzlich wird die Karte mittels Klebestreifen auf das iPad Pro fixiert, um ein Verrutschen zu verhindern.

Um eine Vergleichsbasis herzustellen, werden die gleichen Punkte wie im Kapitel 4.2.3 gemessen. Einziger Unterschied ist eine Anpassung auf Grund der leichten Unterschiede in der Breite der Karte. Die Tabelle 4.3 und Abbildung 4.13 zeigen eine Übersicht über die zu testenden Messpunkte.

#### Versuch

Von den 17 Messpunkten konnten alle fehlerfrei erkannt werden. Dies wurde bei drei durchgeführten Testläufen festgestellt. Eine graphische Übersicht über die Messergebnisse sind der Abbildung 4.14 zu entnehmen.

| Nummer | Soll X | Soll Y | Kommentar |
|---|---|---|---|
| Nr. 1 | 1 | 1 | Eckpunkt links oben |
| Nr. 2 | **26** | 1 | Eckpunkt rechts oben |
| Nr. 3 | 1 | 20 | Eckpunkt links unten |
| Nr. 4 | **26** | 20 | Eckpunkt rechts unten |
| Nr. 5 | **13** | 1 | Mittelpunkt (Mitte, Oben) |
| Nr. 6 | **13** | 20 | Mittelpunkt (Mitte, Unten) |
| Nr. 7 | **13** | 10 | Mittelpunkt (Mitte, Mitte) |
| Nr. 8 | 2 | 1 | Zufallspunkt 1 |
| Nr. 9 | 22 | 5 | Zufallspunkt 2 |
| Nr. 10 | **24** | 8 | Zufallspunkt 3 |
| Nr. 11 | 6 | 5 | Zufallspunkt 4 |
| Nr. 12 | 11 | 2 | Zufallspunkt 5 |
| Nr. 13 | 19 | 15 | Zufallspunkt 6 |
| Nr. 14 | 7 | 18 | Zufallspunkt 7 |
| Nr. 15 | **26** | 20 | Zufallspunkt 8 |
| Nr. 16 | 9 | 17 | Zufallspunkt 9 |
| Nr. 17 | 19 | 19 | Zufallspunkt 10 |

Tabelle 4.3: iPad Pro Demo App: Übersicht der zu messenden Punkte bei den Experimenten. Die Unterschiede zur Tabelle 4.3 wurden fett markiert.

Im Zuge der Tests wurden auch weitere Individualtests durchgeführt. Dabei wurde das Blatt mit einem Raster im Abstand von 0.5cm belegt und somit die zu erkennende Auflösung verdoppelt. Auch bei diesen Tests konnte jede gewählte Position richtig zugeordnet werden.

### 4.3.4 Fazit

Nach der Entwicklung der Demo Applikation und der Durchführung der Experimente können alle im Kapitel 4.1 definierten Anforderungen geprüft werden.

- **Größe des Sensors:** Das iPad Pro besitzt einen Sensor mit der Maße von 26,2 x 19,7cm. Diese Maße entsprichen in etwa einem Blatt DIN-A4 (Die Maße von DIN-A4 im Vergleich dazu: 29,7 x 21cm).
- **Empfindlichkeit:** Da für die Interaktion mit der Karte, diese auf den Sensor gelegt werden muss, spielt die Empfindlichkeit eine wichtige Rolle. Bei den Versuchen konnten bei einem 80 Gramm Papier die Finger problemlos erkannt werden. Ebenso positiv verliefen weitere Tests mit einem 160 Gramm Papier.
- **Genaugigkeit:** Mit einer Genauigkeit von 100% bei der Erkennung der Messpunkte, konnte der Sensors des iPad Pros überzeugen und diese Anforderung optimal

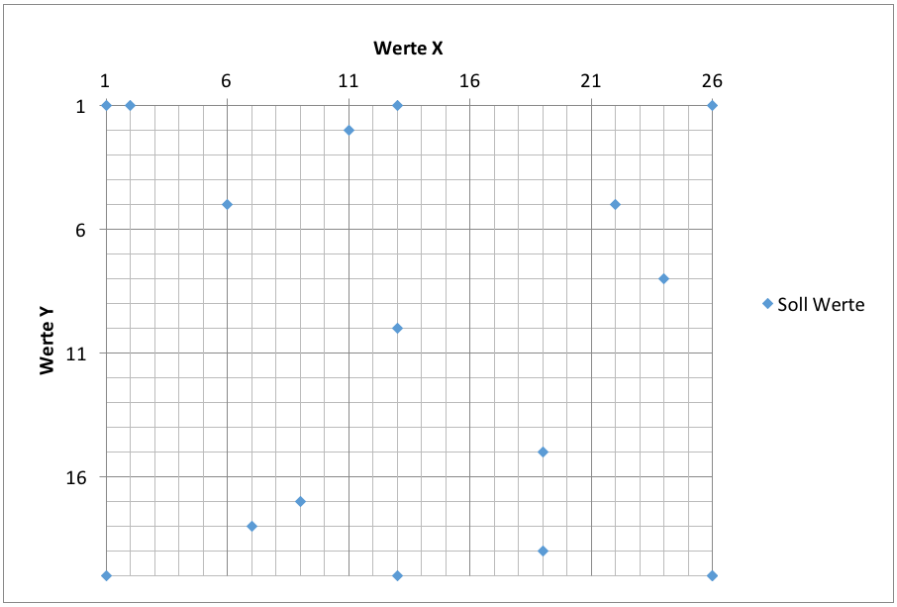


Abbildung 4.13: Graphische Darstellung der gemessenen Punkte.

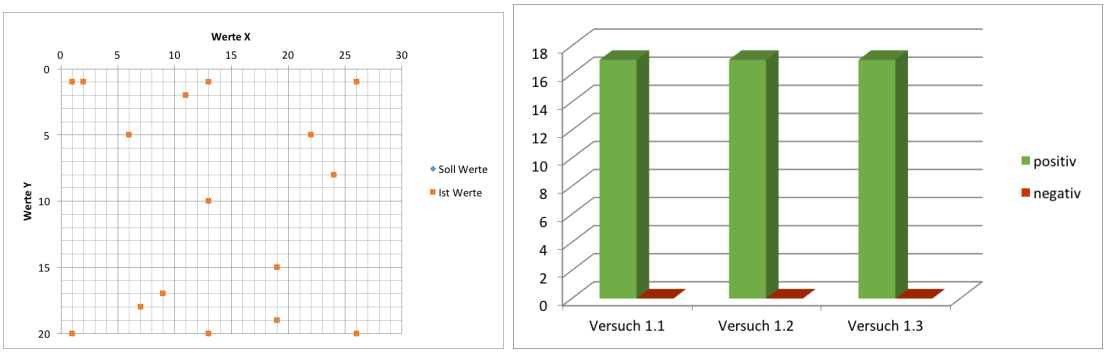


Abbildung 4.14: Visualisierung der Ergebnisse der Messungen: die Darstellung der gemessenen Punkte (links) und die Statistik der drei Testläufe (rechts)

abdecken.

- **Multi-Touch Unterstützung:** Sowohl einzelne, als auch mehrere Finger (Touches) konnten problemlos erkannt werden. Lediglich beim Auflegen der kompletten Handfläche wurde in manchen Fällen, nach dem Entfernen dieser, im Anschluss ein nicht vorhandener Touch erkannt.

- **Gesten Unterstützung:** Die API bieten sieben unterschiedliche Gesten-Erkenner an. Diese können auch mit zahlreichen Kriterien, wie zum Beispiel der Anzahl an

Touches, konfiguriert werden. Des Weiteren können individuelle Gesten-Recognizer selbst implementiert werden.

- **Physischer Aufbau:** Die Karte muss mit dem Sensor verbunden werden, damit diese beim Interagieren nicht verrutscht, und genau an dem Display ausgerichtet liegt. Dafür muss eine Vorrichtung entwickelt werden, um einen Wechsel der Karte zu ermöglichen.
- **Wartung:** Die Wartung des Sensors erfolgt durch das Reinigen des Displays. Die Stromversorgung kann mittels einem iPad Ladekabel auch dauerhaft versorgt werden.
- **API:** Für die Implementierung werden die Programmiersprachen Objective-C und Swift angeboten. Für beide Varianten werden zahlreiche Dokumentationen und Bibliotheken zur Verfügung gestellt.

Im Vergleich zum Leap Motion Sensor ist es bei der Verwendung des iPad Pro ausreichend bei der Software gemäß dem Architekturkonzept von Echtler et al in der Widget-Schicht anzusetzen[EK08]. Jegliches Handling der Gestenerkennung wird einem durch die API des Betriebssystems weitgehend abgenoemmen. Lediglich kleine Eingriffe sind auf der Interpretationsschicht notwendig. Gesamt gesehen bietet das iPad Pro umfassende Möglichkeiten für die Erkennung von Touches, Fingern und Gesten. Die Erkennungsrate liegt mit 100% an oberster Stelle. Die Größe des Sensors ist etwas kleiner als A4, erweist sich aber auch im Praxistest als groß genug, um einen Kartenbereich darzustellen. Einzig für den Aufbau muss ein spezielles Design überlegt werden, damit die Karte immer richtig am Sensor ausgerichtet ist und trotzdem einfach gewechselt werden kann. Angedacht werden kann hierfür eine selbst designte Box aus Holz, Karton oder dem 3D-Drucker.

Eine Implementierung des Prototypen mit dieser Technologie ist möglich und wird angesichts der Möglichkeiten des iPad Pro auch höchstens empfohlen.

## 4.4 Zusammenfassung

In diesem Kapitel wurden die zwei Technologien Leap Motion Sensor und das iPad Pro einzeln, für ihren Protoyp Einsatz, evaluiert. Trotz der starken Unterschiede die beide Technologien zueinander aufweisen, sind beides legitime Ansätze als Basis für diese Art von Konzept. Bei genauerer Betrachtung stellte sich das iPad Pro als Favorit für den im Rahmen dieser Diplomarbeit entwickelten Prototypen heraus. Das iPad Pro konnte sich Dank der fehlerfreien Erkennung von Touches auf der haptischen Karte, gegenüber des etwas ungenaueren Leap Motion Sensors, durchsetzen. Anzumerken ist, dass der Leap Motion Sensor für seine ursprünglich angedachte Verwendungsweise ausreichend genau ist. Lediglich für den alternativen Einsatz, wie hier in diesem Konzept, etwas zu ungenau. Das iPad Pro bietet bereits zahlreiche Funktionen und Bibiliotheken, die für dieses Projekt relevant sind. Viele davon müssten für den Leap Motion Sensor erst entwickelt werden.

Für die Implementierung des Prototypen im Kapitel 5 wird als Basistechnologie das iPad Pro verwendet.

KAPITEL 5

# Implementierung eines Prototypen

Das Kapitel beschäftigt sich mit der Implementierung eines Prototypen, basierend auf den Konzepten von Moser[Mos12] und Sommersguter[Som13]. Diese Konzepte werden im Rahmen der Diplomarbeit zu einem Konzept zusammengeführt, erweitert und anschließend implementiert. Die Kernelemente der Ergebnisse der vorab angeführten Arbeiten sind im Kapitel 2.2 beschrieben. Die für den Prototypen gewählte Technologie ist das iPad Pro (Modell "Late 2015"), das sich bei der Evaluierung im Kapitel 4 als äußerst geeignet herausgestellt hat.

Während der Entwicklung hat sich das Konzept in einigen Punkten mehrfach geändert. Durch die Erkenntnisse der ersten Tests und Versuche beim Testen der Entwicklung, wurden Punkte entdeckt, die durch die ursprünglich definierten Varianten im Konzept nicht zum gewünschten Erfolg führten. Die Punkte wurden im Zuge der Implementierung im Konzept angepasst, so dass diese nun den finalen Zustand der implementierten Applikation entsprechen. Auf Grund der "Henne - Ei" Problematik zwischen der Implementierung und des Konzeptes, wurde es in das Kapitel *Implementierung eines Prototypen* eingegliedert, anstelle eines - wie ursprünglich vorgesehen - separaten Kapitel.

Im ersten Abschnitt 5.1 wird das finale Konzept vorgestellt, das die Grundlage und das Design für die Implementierung der Software- und Hardwarekomponenten darstellt. Das Konzept zeigt die Kernelemente der bisher zum Projekt verfassten Diplomarbeiten, sowie einzelner Anpassungen dieser. Im Abschnitt 5.2 wird die Implementierung der Software beschrieben. Die Aufteilung erfolgt hier pro Kernelement des Konzeptes und zeigt die Umsetzung, sowie den dabei aufgetretenen Schwierigkeiten und deren Abänderungen zum ursprünglichen Design. Das Subkapitel 5.3 beschäftigt sich mit der Aufbereitung der Hardware. Dabei wird auf die Erstellung einer iPad Pro Hülle und der haptischen

Karten eingegangen. Der Abschnitt 5.4 fasst die wichtigsten Punkte des Kapitels noch einmal zusammen.

## 5.1 Finales Konzept der Implementierung

Die Diplomarbeiten von Moser und Sommersguter basieren auf der gleichen Grundidee. Durch das Ausführen von Multi-Touch Gesten auf einer Schwellpapierkarte soll der/die BenutzerIn Audio-Informationen erhalten, bearbeiten und hinzufügen können. Die Karte ist dabei auf einem Multi-Touch Sensor platziert, der für die Gestenerkennung zuständig ist. Beide Diplomarbeiten beschreiben sehr gute Ansätze, die für die Implementierung des Prototypen wiederverwendet werden können. Die Ansätze wurden als Basis für das folgende Konzept herangezogen und um einige neue Aspekte erweitert und angepasst. Gründe dafür sind einerseits die Möglichkeiten, die die gewählten Technologie zur Verfügung stellt, andererseits einige Erkenntnisse bei der Aufarbeitung der Konzepte und der ersten Tests während der Implementierung. Die geänderten Punkte werden bei der Vorstellung des Konzeptes gesondert hervorgehoben und die Hintergründe im Detail dafür besprochen.

### 5.1.1 Applikationszustände

Das Konzept besitzt acht unterschiedlich Zustände, die von der Applikation eingenommen werden können. Jeder Zustand gleicht einem Menüpunkt in der Software und besitzt ein bestimmtes Set an Funktionen. Die verschiedenen Zustände, sowie die möglichen Übergänge dieser, sind in der Grafik 5.4 abgebildet.

#### Start

Der Zustand *Start* ist der Einstiegspunkt der Applikation und wird nach dem Starten, Konfigurieren und Laden der App initiiert. Der/die BenutzerIn bekommt eine Audiomeldung wenn die Applikation vollständig geladen ist und für dem/die BenutzerIn zur Verfügung steht. Der Zustand hat selbst keine Funktionen und dient lediglich als Einstiegspunkt der Software. Aus dem Punkt *Start* können drei weitere Zustände erreicht werden: *Erkundungsmodus*, *Hilfe* und *Einstellungen*. Der Zustand *Start*, *Hilfe* und *Einstellungen* sind die einzigen drei, die auch ohne einer auf dem Sensor liegenden Karte verfügbar sind.

#### Entdeckungsmodus

Der Zustand *Entdeckungsmodus* wird durch das Auflegen einer Karte auf den Sensor aktiviert. Er ist dafür vorgesehen die Karte mit den Händen zu erkunden. In diesem Zustand werden keinerlei Informationen, Geräusche oder Töne über die Applikation ausgegeben. Durch das Entfernen der Karte wird der Entdeckungsmodus verlassen und die Applikation kehrt in den Zustand *Start* zurück. Mit Hilfe von diversen Gesten können die Zustände *Ort*, *Ebenen*, *Gespeicherte Orte* und *Distanz* erreicht werden.

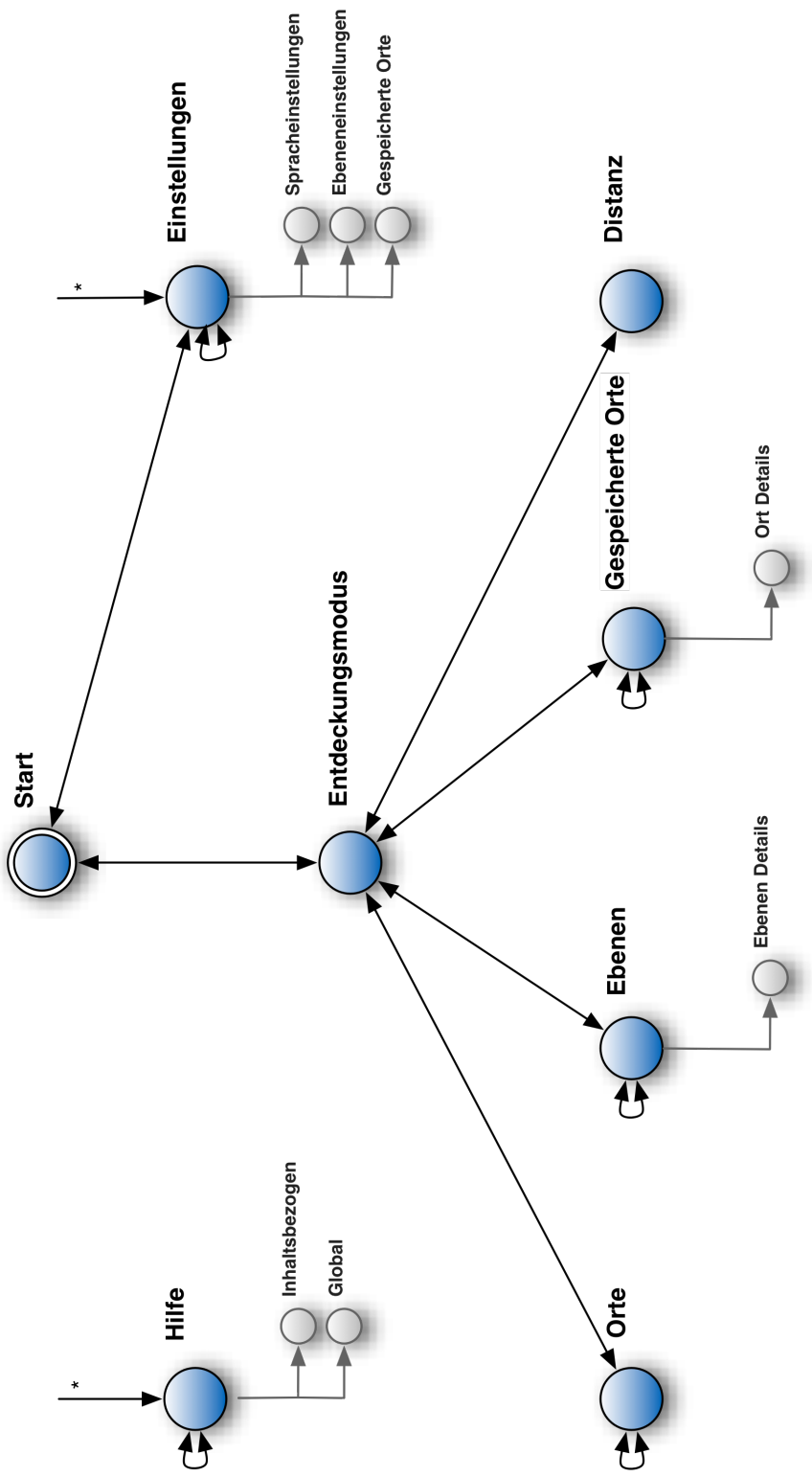


Abbildung 5.1: Übersicht über alle möglichen Zustände der Applikation und deren Übergänge. Der Punkt Start ist der Einstiegspunkt in die Applikation. Die Punkte Hilfe und Einstellungen können aus jedem Zustand erreicht werden.

### Ebenen

Eine Ebene ist eine Gruppierung von geobasierten Informationen die zu einem Thema zusammengefasst sind. Jede Ebene besitzt dabei eine beliebige Anzahl an Informationen, die für die aktuelle Karte relevant sind. Sowohl die Themen der Ebenen, als auch die Inhalte sind von Karte zu Karte unterschiedlich. Typische Ebenen sind zum Beispiel: Straßennamen, Öffentliche Gebäude, Transportmittel und Parks. Ist eine Ebene aktiviert, wird bei der Auswahl eines Punktes auf der Karte beim Erkunden, die hinterlegten Informationen dieser Ebene als Quelle für die Ausgabe herangezogen. Deaktivierte Ebenen werden dabei nicht berücksichtigt. Im Zustand *Ebenen* kann der/die BenutzerIn alle, für diese Karte selektierten, Ebenen durchblättern und diese für die Ausgabe aktivieren oder deaktivieren. Ebenen die bereits in den Einstellungen deaktiviert wurden, stehen dem/der BenutzerIn in diesem Zustand nicht mehr zur Verfügung. Neben der De- und Aktivierung der Ebenen, können auch alle Informationen dieser Ebene ausgegeben werden. Dieser Punkt soll dem/der BenutzerIn die Suche nach bestimmten Orten, Geschäften oder anderen Punkten von Interesse erleichtern.

### Ort

Auf der fühlbaren Karte werden Orte, die für den/die BenutzerIn von besonderem Interesse sein können, mit Informationen hinterlegt. Wird einer dieser Punkt im Erkundungsmodus zur Ausgabe von Informationen ausgewählt, wird der Zustand *Ort* erreicht. Im Zustand *Ort* wird die Ausgabe aller zum gewählten Punkt hinterlegten Informationen aktiviert. Dabei werden nur Informationen ausgegeben, die in den aktivierten Ebenen enthalten sind. Informationen aus deaktivierten Ebenen werden nicht berücksichtigt. Alle Informationen werden dabei als Liste angeordnet und automatisch einzeln nacheinander ausgegeben. Zusätzlich kann der/die BenutzerIn einzelne Informationen überspringen oder wiederholen. Möchte sich ein/eine BenutzerIn den gewählten Ort merken, kann der Punkt für später gespeichert werden. Dabei wird der Ort, inklusive aller zu diesem Zeitpunkt verfügbaren Informationen, gespeichert.

### Gespeicherte Orte

Im Zustand *Gespeicherte Orte* kann der/die BenutzerIn alle von ihm gespeicherten Orte einsehen. Dabei werden nur die gespeicherten Orte, die sich auf der aktuellen Karte befinden, ausgegeben. Die Wiedergabe der Informationen ist dabei unabhängig von der Auswahl an aktivierten Ebenen. Diese entspricht den gleichen Informationen wie zum Zeitpunkt des Speicherns und den zum damaligen Zeitpunkt aktivierten Ebenen.

### Distanz

Im Zustand *Distanz* wird die Entfernung zweier gewählter Punkte berechnet und dem User ausgegeben. Die Entfernung wird dabei in Metern angegeben, da der Großteil der Karten nur einen sehr kleinen Bereich eines Gebietes abbilden und hier die Angabe in Metern verständlicher ist als Kilometerangaben mit Kommastellen. Karten die ganze

Bundesländer oder Staaten abbilden, werden in diesem Konzept nicht berücksichtigt. Eine Angabe von Wegzeiten ist nicht erwünscht, da diese je nach Stärke der Sehbehinderung und Alter der Personen unterschiedlich sein können.

#### Hilfe

Im Zustand *Hilfe* werden dem/der BenutzerIn Informationen über die Verwendung der Applikation, deren Zustände und möglichen Gesten zur Verfügung gestellt. Dieser Zustand kann aus jedem beliebigen Zustand geöffnet werden. Es kann dabei zwischen zwei unterschiedliche Arten von Informationen gewählt werden. Der Menüpunkt *Global* gibt Informationen über die Applikation, deren Hintergrund und eine Liste aller Zustände und Gesten aus. Die Informationen sind dabei in einer Liste angeordnet und können individuell nach vorne und hinten weitergeblättert werden. Der zweite Menüpunkt *Inhaltsbezogen* gibt Informationen über den jeweiligen Zustand und dessen mögliche Funktionen und Gesten aus, aus dem die Hilfe aufgerufen wurde. Auch diese Informationen sind als Liste angeordnet und navigierbar.

#### Einstellungen

Im Modus *Einstellungen* können diverse Anpassungen der Applikation durch den User vorgenommen werden. Dabei wird zwischen drei Kategorien unterschieden:

- In den *Spracheinstellungen* ist das Sprechtempo regulierbar. Der Standardwert liegt dabei bei 100% und kann in einem Bereich von 50% bis 150% in Zehnerschritten variiert werden. Durch einen niedrigeren Wert wird das Tempo langsamer, während bei einem höheren Wert das Sprechtempo steigt.

- Im Menüpunkt *Ebeneneinstellungen* können Ebenen Global aktiviert oder deaktiviert werden. Ist eine Ebene aktiviert wird diese für die Ausgabe der Informationen zu Punkten auf der Karte herangezogen. Im deaktivierten Zustand stehen Informationen dieser Ebene nicht zur Verfügung. Diese Einstellung ist unabhängig von den Karten, die für die Erkundung ausgewählt werden, und somit für die gesamte Dauer der Erkundung bis zum Beenden der Applikation gültig. Standardmäßig sind nach dem Starten der Applikation alle Ebenen aktiviert.

- Im Punkt *Gespeicherte Orte* stehen dem/der BenutzerIn alle, seit dem Start der Applikation gespeicherten Orte, zur Verfügung. Alle gespeicherten Orte sind nach dem Zeitpunkt des Speicherns sortiert, in einer Liste abgelegt und können einzeln abgerufen werden.

Die Implementierung der Einstellungen soll generisch erfolgen, um diese später bei Bedarf um zusätzliche Einstellungen erweitern zu können.

#### Anpassungen zu den Konzept von Sommersguter und Moser

Im Vergleich zu den Basiskonzepten sind folgende Punkte angepasst worden:

- **Hilfe:** Im Ursprungskonzept konnte der Zustand *Hilfe* nur aus dem Initialzustand *Start* erreicht werden. Dieses Konzept wurde so erweitert, dass die Hilfe aus jedem Zustand erreichbar ist, um so dem/der BenutzerIn die Möglichkeit nach mehr Information bieten zu können. Zusätzlich wurde der Punkt *Hilfe* in zwei Bereiche unterteilt. Der erste Unterpunkt bietet globale Informationen, während der zweite Unterpunkt spezifische Informationen über den Zustand gibt, aus dem die Hilfe aufgerufen wurde. Dies bietet neuen BenutzerInnen mehr Hilfestellungen für die Benützung der Applikation.
- **Eigene Orte**: Im Konzept von Sommersguter und Moser war vorgesehen, dass es eine Möglichkeit gibt, um eigene Informationen an Orten zu hinterlegen. Dieser Punkt wurde auf Grund der ansonsten zu stark wachsenden Komplexität verworfen. Stattdessen können gespeicherte Orte aus der Erkundung, wie auch bereits im Zustand *Einstellung*, aufgerufen und wiedergegeben werden. Ein großer Vorteil ist, dass nun der/die BenutzerIn die Karte, für das Aufrufen der gespeicherten Orte, nicht entfernen muss.
- **Einstellungen:** Wie auch bereits bei der Hilfe, ist auch der Zustand *Einstellungen* aus jedem Punkt aus erreichbar. Hintergrund dafür sind die Einstellungen der Sprachgeschwindigkeit, die dem/der BenutzerIn zu jeder Zeit zur Verfügung gestellt sind, ohne die Karte entfernen zu müssen.

### 5.1.2 Gesten

Die Interaktion der BenutzerInnen mit der Applikation erfolgt durch ein Set von Gesten und ist eines der wichtigsten Instrumente zur Steuerung. Aus diesem Grund wird hier ein besonderes Augenmerk auf die Definition, Eindeutigkeit sowie die Bedeutung der Gesten gelegt.

In diesem Konzept sind sieben Basisgesten definiert, die durch unterschiedlichste Ausführungen das gesamte Set für die Interaktion darstellen. Die Basisgesten, sowie deren Varianten, werden nun im Detail vorgestellt. Abbildung 5.2[1] zeigt eine grafische Darstellung aller Gesten.

#### Tap

Die Geste *Tap* ist ein einfaches kurzes Berühren der Oberfläche. Diese Berührung dauert dabei kürzer als 100ms[Saf08].

Diese Geste ist alleinstehend nicht in der Applikation verankert, wird aber für die Beschreibung der nachfolgenden Gesten benötigt.

[1] Die Basisgrafiken unterliegen dem (CC) BY-SA 3.0, Wolfgang Maehr

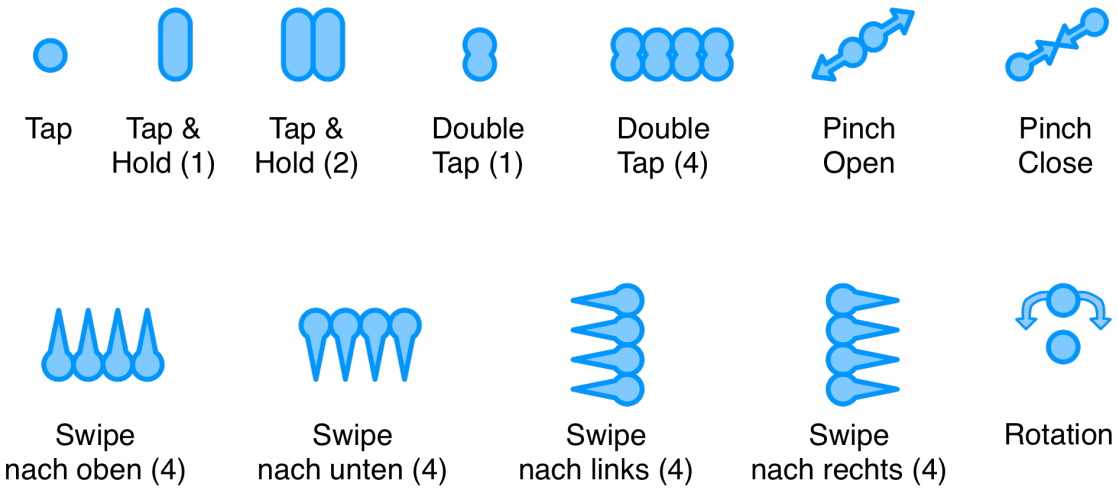


Abbildung 5.2: Übersicht über alle Gesten, die im Rahmen des Konzeptes für die Interaktion mit der Software definiert wurden.

#### Tap & Hold

Diese Geste beschreibt eine Variante zur Geste *Tap*. Im Vergleich zum *Tap* wird die Oberfläche nicht nur für einen kurzen Zeitpunkt berührt, sondern der Finger ruht eine unbestimmte Zeit (mindestens aber 100ms) auf diesem Punkt.

Diese Geste kann mit einem und zwei Fingern ausgeführt werden. Je nach Anzahl werden dem/der BenutzerIn unterschiedliche Informationen zur Verfügung gestellt.

- **Tap & Hold (1):** Ausgabe der Informationen zum gewählten Punkt auf der Karte. Solange diese Geste gehalten wird, werden dem/der BenutzerIn Informationen via Audio zur Verfügung gestellt.
- **Tap & Hold (2):** Ausgabe der Distanz zwischen den gewählten Punkten.

#### Double Tap

Der *Double Tap* ist die zweifache Ausführung eines *Taps* in einem sehr kurzen Abstand. Dieser Abstand beträgt dabei weniger als 75ms[Saf08]. Die Ausführung dieser Geste kann mit einer unterschiedlichen Anzahl an Fingern durchgeführt werden. Im Zuge dieser Realisierung sind folgende Varianten definiert:

- **Double Tap (1):** In dieser Variante wird die Geste mit einem Finger ausgeführt. Der Grundgedanke dieser Geste besteht darin, dass Elemente selektiert und ausgewählt werden können. Bei der Navigation wird durch einen *Double Tap (1)* der aktuelle Menüpunkt ausgewählt und der dazugehörige Zustand betreten. Zusätzlich

wird diese Geste zum Aktivieren der Detailansicht von Elementen in den Zuständen *Ebenen* und *Gespeicherte Orte* verwendet.

- **Double Tap (4):** Die Ausführung der Geste mit vier Fingern gibt den Namen des Zustandes aus, in dem sich der/die BenutzerIn aktuell befindet. Dies bietet dem/der BenutzerIn zu jedem Zeitpunkt die Möglichkeit, den eigenen Standort in der Applikation zu erfragen.

#### Triple Tap

Analog zum *Double Tap* ist diese Geste die dreifache Ausführung eines *Taps* in einem kurzen Intervall. In diesem Konzept ist die Geste nur als *Triple Tap (4)* vorgesehen. Bei der Ausführung eines dreifachen Taps mit vier Fingern wird der zum Zustand gehörige Hilfetext ausgegeben. Ziel dieser Geste ist es, einen schnellen Überblick über die Möglichkeiten im aktuellen Zustand zu bekommen, ohne den Zustand zu verlassen.

#### Pinch Open / Pinch Close

Die Geste *Pinch Open* beschreibt zwei nebeneinander liegende Finger, die sich voneinander entfernen. Die Geste *Pinch Close* sind zwei auseinander liegende Finger, die sich auf einander zu bewegen. Diese Gesten werden in der Applikation in nur einer Variante, mit zwei Fingern *Pinch Open (2) / Pinch Close (2)*, durchgeführt.

Die Bedeutung *Close* entspricht dem Schließen eines Menüs oder Zustandes, während *Open* das Öffnen beschreibt. Die Geste *Pinch Close (2)* wird als universelle Geste für das Schließen des aktuellen Zustandes beziehungsweise dem Zurückgehen eingesetzt. Der/die BenutzerIn kann durch das Ausführen dieser Geste den jeweiligen Zustand verlassen und in den vorigen Zustand zurückkehren. Die Geste *Pinch Open (2)* wird zum Öffnen der Zustände *Einstellungen* und *Layers* verwendet.

#### Swipe Links / Swipe Rechts / Swipe nach oben / Swipe nach unten

Die Geste *Swipe* beschreibt das gleichzeitige Bewegen von Fingern in ein und dieselbe Richtung. Die Geste selbst kann mit einer beliebigen Anzahl von Fingern durchgeführt werden, wobei für dieses Konzept die Anzahl auf vier Fingern limitiert ist. Die Bedeutung der Geste ist abhängig von der Richtung der Ausführung. Diese Bedeutungen sind für die gesamte Applikation gleich definiert, um dem/der BenutzerIn ein durchgängiges Interaktionskonzept zu bieten.

Durch die Ausübung der Geste *Swipe Links* wird das Zurückblättern in einer Liste ausgedrückt. Im Gegenteil dazu symbolisiert die Geste *Swipe Rechts* das Weiterblättern in einer Liste. Durch die Gesten *Swipe nach oben* und *Swipe nach unten* können diverse Inhalte je nach Zustand entweder aktiviert/deaktiviert oder gespeichert/entfernt werden.

### Rotor

Die Geste *Rotor* entspricht einer im Kreis angeordneten Navigation. Die Startposition beginnt mit der vertikalen Ausrichtung von Daumen und Zeigefinger am Sensor. Durch die Bewegung des Zeigefingers nach links und rechts werden die einzelnen Elemente der Navigation ausgewählt. Der Daumen bewegt sich dabei nicht und stellt die untere Kante des Kreises dar, während der Zeigefinger sich entlang des Umfanges bewegt. Die selektierten Punkte werden durch das Loslassen der Geste gewählt. Die Grundidee dieser Geste ist dem VoiceOver Rotor Konzept von Apple entnommen, das mit der iOS Version 8 eingeführt wurde[Appd]. Der Kreis ist in drei Segmente unterteilt. Der linke Teil ruft den Zustand Hilfe auf, während der rechte Teil die Einstellungen auswählt. Der Bereich in der Mitte ist neutral und wird als Abbruch der Geste ohne einer Aktion gewertet.

Die Abbildung 5.3 zeigt eine grafische Darstellung des Rotor-Konzeptes inklusive Beispielen für die Auswahl der unterschiedlichen Bereiche.

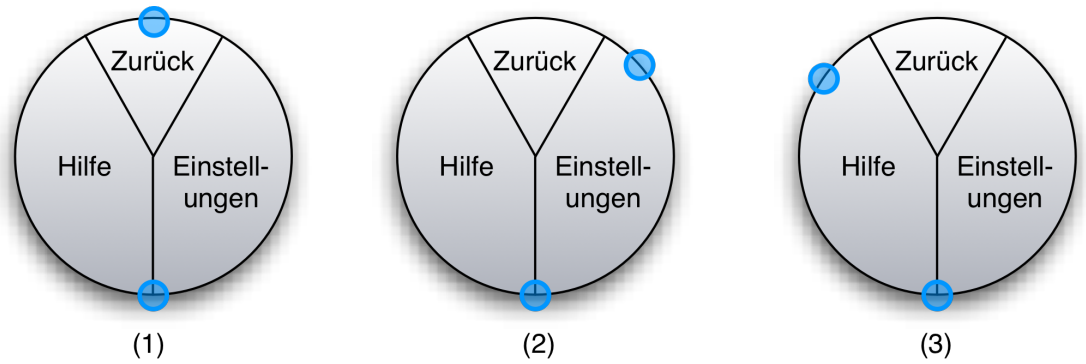


Abbildung 5.3: Grafische Darstellung des Rotor Konzeptes. Die blauen Kreise in der Grafik stellen die Touch Positionen am Sensor dar. Durch das Bewegen des oberen Touchpunktes können unterschiedliche Bereiche des Rotors durch das Loslassen der Touchpunkte selektiert werden. Abbildung (1) zeigt die Ausgangsposition, die den Bereich *Zurück* selektiert, während die Abbildung (2) den Bereich *Einstellungen* und Abbildung (3) den Zustand *Hilfe* ausgewählt hat.

### Anpassungen zum Konzept von Sommersguter

Im Zuge der Implementierung wurde damit begonnen, die definierten Gesten im Konzept von Sommersguter identisch umzusetzen. Bei der Durchführung der ersten Tests sind Punkte aufgekommen, die Anpassungen des Konzeptes bei den Definitionen der Gesten erforderten. Zusätzlich wurden zu den durchgeführten Tests auch Workshops zur Definition der neuen Gesten durchgeführt.

- **Swipe (4):** Die Geste *Swipe* ist in diesem Konzept für die Navigation von Einträgen in einer Liste vorgesehen. Dabei ist es irrelevant ob es sich um eine Liste mit

Menüeinträgen oder Inhalten handelt. Von Sommersguter war vorgesehen, dass diese Geste mit fünf (5) Fingern ausgeführt wird. Die ersten Tests haben gezeigt, dass die Erkennungsrate bei fünf Fingern sehr niedrig ist und oftmals als *Pinch Open* oder *Pinch Close* erkannt wird. Durch die Abänderung auf vier (4) Finger konnte die Erkennungsrate gesteigert werden.

- **Rotor:** Wie bereits im Abschnitt 5.1.1 vorgestellt, sollen die Punkte *Hilfe* und *Einstellungen* jederzeit aufrufbar sein. Um dieses Ziel zu erreichen, muss dafür eine eigene Geste vorgesehen werden, die sich mit keiner anderen Geste überlagert und zu Fehlverhalten führen könnte. Bei der Evaluierung der VoiceOver Funktionen des Apple iOS wurde das Rotor Konzept entdeckt und für diese Zwecke implementiert. Aktuell sind drei Menüpunkte im Rotor vorgesehen.
- **Double Tap (4):** Für Personen ohne Sehbeeinträchtigungen ist es relativ einfach sich in einer Applikation zurecht zu finden, da oftmals zahlreiche visuelle Hilfestellungen über den aktuellen Zustand oder den ausgewählten Menüpunkt geboten werden. Als blinde Person kann es sehr schnell zu Verwirrungen kommen, besonders wenn Gesten falsch von der Applikation erkannt werden. Aus diesem Grund kann mit Hilfe der Geste *Double Tap (4)* jederzeit der Name des aktuellen Zustandes ausgegeben werden. Zusätzlich wird beim Wechseln des Zustandes der Name des neuen Zustandes ausgegeben. Durch diese zwei Anpassungen soll dem/der BenutzerIn die Orientierung erleichtert werden.
- **Triple Tap (4):** Die Geste *Triple Tap (4)* ist, wie bereits auch die Geste *Double Tap (4)*, als Hilfestellung für den/die BenutzerIn vorgesehen. Durch die Ausführung dieser Geste wird der Hilfetext für den aktuellen Zustand ausgegeben. Diese Hilfe bietet einen Überblick über alle möglichen Gesten und deren Bedeutung in diesem Zustand.

### 5.1.3 Interaktionskonzept

Das Interaktionskonzept beschreibt die Kombination der Gesten und Zustände. Dabei wird erläutert, welche Funktionen die Gesten in jedem Zustand haben und wie damit zwischen den Zuständen navigiert werden kann. Die Abbildung 2.1 zeigt eine Darstellung aller im Kapitel 5.1.1 definierten Zustände und erweitert diese um jeweils eine Geste pro Übergang. Das Symbol "»" kennzeichnet dabei die Geste zum Betreten des Zustandes, während das Symbol "«" die Aktion zum Verlassen des Zustandes beschreibt. Die Gesten sind so gewählt, dass diese soweit wie möglich ein durchgängiges Konzept für den/die BenutzerIn bieten. Zur Vereinfachung des Konzeptes soll eine globale Geste für das Verlassen und Schließen implementiert werden, die zu jederzeit ausgeführt werden kann. Damit erschließt sich die Möglichkeit zu jeder Zeit den aktuellen Zustand zu verlassen, um in einen übergeordneten Zustand zurückkehren zu können.

Zusätzlich zu den Übergängen beschreibt das Interaktionskonzept auch die Gesten, die zur Navigation in den jeweiligen Zuständen vorgesehen sind. Die Tabelle 5.1 zeigt eine Übersicht, nach Zustand gruppiert, über alle möglichen Gesten und deren Bedeutungen.

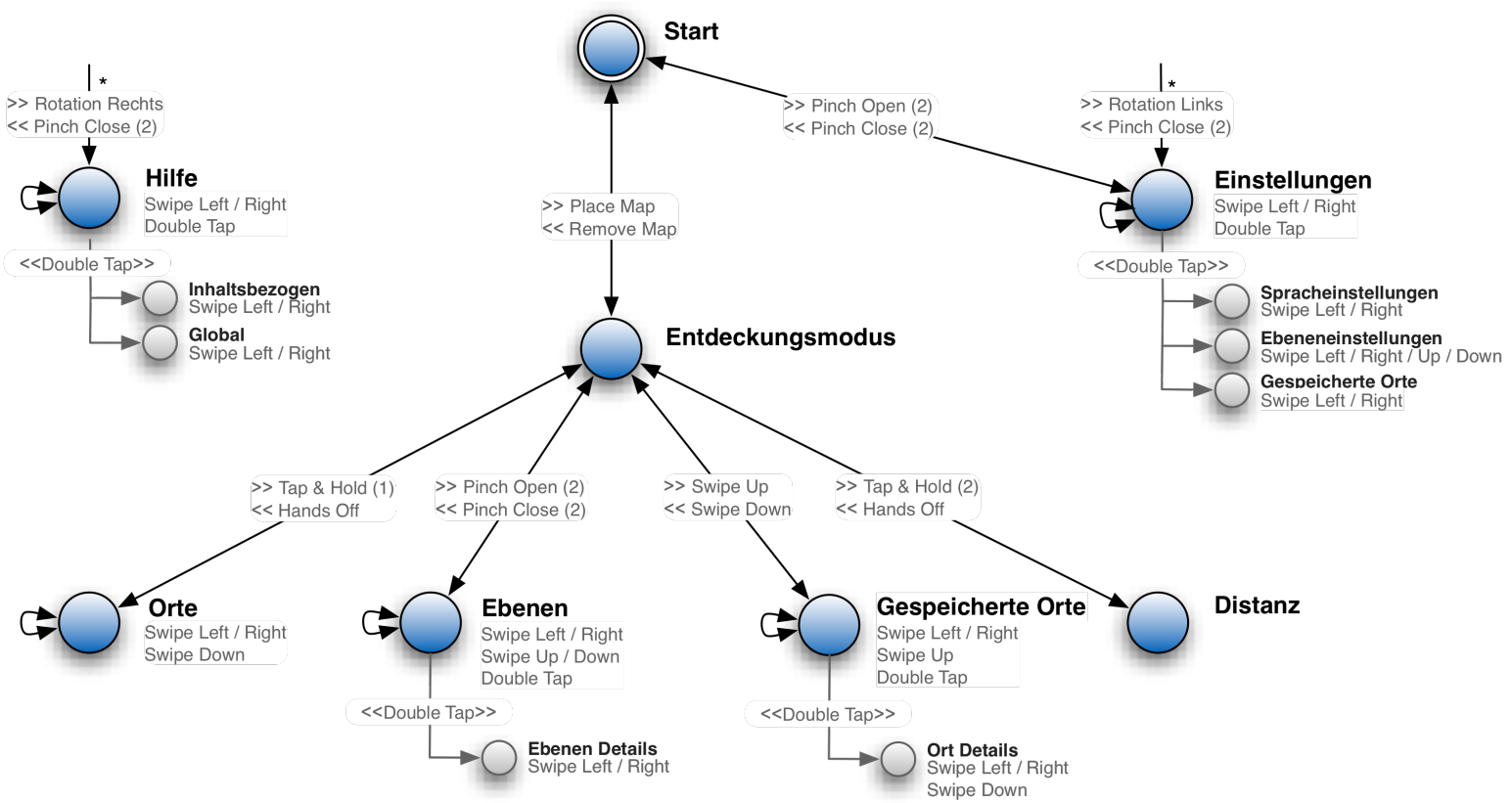


Abbildung 5.4: Darstellung aller möglichen Zustände der Applikation und der jeweils definierten Gesten als Übergänge. Die Pfeile markieren die möglichen Übergänge zwischen den Zuständen. Das Symbol "»" kennzeichnet dabei die Geste zum Betreten des Zustandes, während das Symbol "«" die Aktion zum Verlassen des Zustandes beschreibt. Der Punkt *Start* ist der Einstiegspunkt in die Applikation. Die Punkte *Hilfe* und *Einstellungen* können aus jedem Zustand erreicht werden.

| Geste | Bedeutung |
|---|---|
| **Globale Gesten** | |
| Pinch Close (2) | Universelle Geste zum Beenden des aktuellen Zustandes-/Unterpunktes. |
| Rotation Links | Aufrufen des Zustandes Hilfe. |
| Rotation Rechts | Aufrufen des Zustandes Einstellungen. |
| Rotation Mittig | Keine Aktion. |
| **Ort** | |
| Swipe Links (4) | Eine Information wiederholt anhören oder zurück springen. |
| Swipe Rechts (4) | Eine Information weiterschalten/überspringen. |
| Swipe nach unten (4) | Den gewählten Ort speichern. |
| **Ebenen** | |
| Swipe Links (4) | Eine Ebene zurück blättern. |
| Swipe Rechts (4) | Eine Ebene weiter blättern. |
| Swipe nach oben (4) | Aktivieren der gewählten Ebene. |
| Swipe nach unten (4) | Deaktivieren der gewählten Ebene. |
| Double Tap (1) | Auswahl der Ebene |
| **Ebenen - Details** | |
| Swipe Links (4) | Eine Information wiederholt anhören oder zurück springen. |
| Swipe Rechts (4) | Eine Information weiterschalten/überspringen. |
| **Gespeicherte Orte** | |
| Swipe Links (4) | Einen Ort zurück blättern. |
| Swipe Rechts (4) | Eine Ort weiter blättern. |
| Swipe nach oben (4) | Löschen des gewählten Ortes. |
| Double Tap (1) | Auswahl des Ortes |
| **Gespeicherte Orte - Details** | |
| Swipe Links (4) | Eine Information wiederholt anhören oder zurück springen. |
| Swipe Rechts (4) | Eine Information weiterschalten/überspringen. |
| **Einstellungen** | |
| Swipe Links (4) | Im Menü/Liste nach Links navigieren |
| Swipe Rechts (4) | Im Menü/Liste nach Rechts navigieren |
| Double Tap (1) | Auswahl des Menüpunktes |
| **Einstellungen - Ebenen** | |
| Swipe nach oben (4) | Aktivieren der gewählten Ebene |
| Swipe nach unten (4) | Deaktivieren der gewählten Ebene |
| **Hilfe** | |
| Swipe Links (4) | Im Menü/Liste nach Links navigieren |
| Swipe Rechts (4) | Im Menü/Liste nach Rechts navigieren |
| Double Tap (1) | Auswahl des Menüpunktes |

Tabelle 5.1: Übersicht über die definierten Gesten pro Zustand und deren Bedeutung in der Applikation.

**Anpassungen zum Konzept von Sommersguter**

Die Änderungen zum ursprünglichen Konzept, sowie die Hintergründe der Veränderungen werden in den folgenden Punkten erklärt:

- **Adaptionen der Gesten:** Betrachtet man das Interaktionskonzept von Sommersguter im Abschnitt 2.2 werden einige Änderungen der Gesten aufgezeigt. Die Geste *Pinch Open (5)* wurde zu *Pinch Open (2)* abgeändert, da dies im Betriebssystem iOS standardmäßig unterstützt wird. Da keine weiteren Varianten des *Pinch Open* mit einer anderen Fingeranzahl vorgesehen war, stellte sich dies als unproblematisch heraus. Aus dem gleichen Grund wurde die Geste *Pinch Close (5)* zu *(2)* abgeändert. Durch die Umgestaltung des Zustandes *Eigene Orte* zu *Gespeicherte Orte* wurde auch die dazugehörige Geste entsprechend angepasst. Ein Ort wird durch die Geste *Swipe nach unten (4)* gespeichert. Dies entspricht der Denkweise des Menschen "etwas zu sich her zu ziehen". Aus diesem Grund wurde das Anzeigen der gespeicherten Orte vice versa mit der Geste *Swipe nach oben (4)* - etwas von sich weg schieben - hinterlegt.
- **Universal Closure:** Um die Gestenvielfalt für den/die BenutzerIn übersichtlich zu gestalten, wurde eine Geste für das Beenden von Zuständen oder dem Zurück in der Navigation einheitlich definiert. Diese Geste zum universellen Schließen wurde mit *Pinch Close(2)* festgelegt und ist aus jedem Zustand ausführbar.
- **Rotor:** Eine komplett neue Geste wurde mit dem Konzept des Rotors eingeführt. Der Rotor dient dazu, die Punkte *Hilfe* und *Einstellungen* aus jedem Zustand aus aufrufen zu können. Dies war im ursprünglichen Konzept von Sommersguter nicht vorgesehen. Im weiteren Verlauf stellte sich diese Geste als sehr benutzerfreundlich heraus, da beispielsweise die Sprachgeschwindigkeit auch während der Erkundung geändert oder der Menüpunkt *Hilfe* aufgerufen werden kann.
- **Double Tap (4)/ Triple Tap (4):** Während der ersten Tests durch außenstehende Personen sind zwei Fragen aufgekommen: "Wo befinde ich mich? Was kann ich jetzt tun?". Um diese Fragen zu beantworten, wurde das Konzept um zwei weitere Gesten erweitert, um dem/der BenutzerIn Hilfestellungen zu geben. Durch das Ausführen der Geste *Double Tap (4)* wird der Name des aktuellen Zustandes ausgegeben. Durch das Ausführen der Geste *Triple Tap (4)* wird der Hilfetext des aktuellen Zustands aufgerufen. Durch diese zwei Gesten lassen sich die essentiellen Fragen der BenutzerInnen beantworten.

### 5.1.4 Sprachausgabe

Menschen mit einer Sehbeeinträchtigung sind auf die Führung durch die Software und dessen Navigation anhand der Ausgabe von Informationen via Audio angewiesen. Aus diesem Grund ist die Sprachausgabe ein wichtiges Kernelement des Konzeptes. Als Unterstützung sollen dem/der BenutzerIn alle Aktionen als Sprachtext wiedergegeben

werden. Für die Sprachausgabe wird die im Apple iOS integrierte *Text-to-Speech*[2] Funktion verwendet[SW14]. Notwendige Einstellungen, wie zum Beispiel die Sprachgeschwindigkeit, sind aus der Applikation heraus zu konfigurieren. Die Ausgabe der Sprache selbst wird über die verbauten Lautsprecher des iPad Pro ausgegeben. Die Lautstärke kann über die seitlich am iPad Pro angebrachten Regler eingestellt werden.

Alle Texte die in der Applikation in Verwendung sind, werden als Textbausteine definiert, um eine schnelle und einfache Wartung zu gewährleisten. Diese sollen als XML-Datei in der Software oder in einer Datenbank abgelegt werden. Ein weiterer Vorteil neben der Wartbarkeit ist, dass für die Änderung der Ausgabesprache in der Applikation lediglich die Textbausteine übersetzt werden müssen. Eine Änderung des Software Codes ist dafür nicht notwendig.

Zusätzlich sollen neben der Ausgabe von gesprochenen Audioinformationen dem/der BenutzerIn auch Töne ausgegeben werden, um die Navigation zu erleichtern[WNL06]. Diese Töne werden Earcons genannt. Jeder Ton soll eindeutig einer Situation zugewiesen sein. Diese Töne werden zum Beispiel abgespielt, wenn das Ende oder der Anfang einer Liste erreicht ist oder wenn eine Information einer neuen Ebene ausgegeben wird. Durch das Signalisieren vom Ende einer Information wird dem/der BenutzerIn mitgeteilt, dass keine weiteren Informationen verfügbar sind.

#### Anpassungen zum Konzept von Moser

Im Gegensatz zum Konzept von Moser wird in dieser Implementierung nicht mit Audioinformationen als MP3-Dateien gearbeitet. Stattdessen werden die Informationen als Text hinterlegt und mit Hilfe einer Text-to-Speech Funktion in gesprochene Sprache umgewandelt. Durch diesen Ansatz können Informationen einfacher für die Verwendung in der Applikation herangezogen werden, ohne diese vorab als MP3-Dateien aufzunehmen.

Ein Vorteil für den/die BenutzerIn ergibt sich durch die Flexibilität bei der Einstellung der Sprachgeschwindigkeit. Jede Person kann die Einstellung individuell auf sich abstimmen, um so ein vertrautes Umfeld herzustellen. Folgendes ergibt sich durch die häufige Benutzung der Text-to-Speech Funktion, welche aus anderen Anwendungsfällen bei der Computernutzung bereits bekannt ist. Daraus ableitend lässt sich festhalten, dass die Klangfarbe der Stimme sowie die Sprachgeschwindigkeit zu einer effizienten Nutzbarkeit der Applikation beitragen.

### 5.1.5 Kartenerkennung

Im Konzept von Moser wurden zwei Varianten zur Kartenerkennung vorgestellt. Eine optische Variante und und eine Variante mit dem Einsatz eines RFID-Chips. Das iPad Pro bietet standardmäßig kein Lesegerät um RFID- oder NFC-Chips zu erkennen und auszulesen. Aus diesem Grund wird die Variante der optischen Erkennung gewählt. Das

[2]Text-to-Speech, auf Deutsch: Text zu Sprache ist eine Software Komponente, die Text in Sprache umwandelt und ausgibt.

iPad Pro besitzt auf der Frontseite eine Kamera, die es ermöglicht Fotos und Videos zu erstellen. Diese Kamera soll für die Erkennung der Karten verwendet werden. Der Bereich der Karte, der beim Aufliegen auf dem Sensor die Kamera überdeckt, wird dabei herausgeschnitten und durch ein farbiges Papier ersetzt. Durch das Erstellen von Fotos in regelmäßigen Intervallen und dem Auswerten der Farbwerte der Bilder, sollen die durch unterschiedliche Farben markierten Karten erkannt werden. Abbildung 5.5 zeigt wo diese Farbpunkte - in diesem Beispiel die Farbe Grün - auf den Karten angebracht werden.

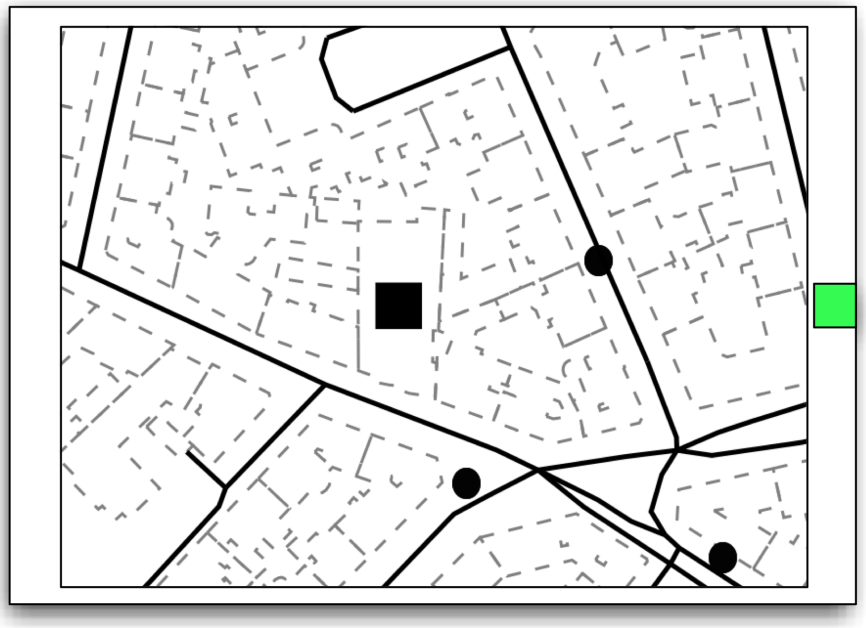

Abbildung 5.5: Die haptischen Karten werden im Bereich der Kamera des iPad Pro mit einem Farbigen Papier bestückt. Die Erkennung der Karte erfolgt durch das Auslesen der Farbwerte der von der Kamera aufgenommenen Fotos.

### Anpassungen zum Konzept von Moser

Die, wie im Konzept von Moser vorgestellte, Verwendung von Strich- oder QR-Codes ist durch die Verwendung des iPad Pro nur bedingt verwendbar. Zur Erkennung dieser Codes müsste der Code der Karte in einem Abstand, in dem die Kamera ausreichend fokussieren kann, über die Kamera gehalten werden. Durch diese Voraussetzung ist die Erkennung nur dann möglich, wenn die Person weiß wo sich die Kamera befindet und in etwa weiß wie hoch die Karte gehalten werden muss. Das automatische Erkennen über das Entfernen der Karte ist dadurch nicht gegeben, da der Code nur einmal initial gelesen wird. Ein regelmäßiger Check ist durch das notwendige hochhalten der Karte nicht möglich. Die Person müsste, durch zum Beispiel einer Geste, der Applikation das Abnehmen der Karte mitteilen.

### 5.1.6 Datenkonzept

Das Datenkonzept beschreibt die Verwaltung aller für die Applikation notwendigen Daten. Dabei gibt es unterschiedliche Daten, die dabei berücksichtigt werden müssen:

- **Konfigurationen der Karten:** Für die Verwendung der Karten müssen diese konfiguriert und mit Daten hinterlegt werden. Die Koordinaten der Eckpunkte markieren den Bereich der Karte und werden von der Software zur Umrechnung der Touchpunkte auf Koordinaten verwendet. Jede Karte wird beim Konfigurieren mit einer Information über den dargestellten Bereich der Karte befüllt, die dem/der BenutzerIn beim Auflegen dieser auf dem Sensor wiedergegeben wird.

- **Konfigurationen der Ebenen:** Alle Ebenen, die in der Applikation verfügbar sein sollen, werden initial angelegt und mit Daten versorgt. Die Ebenen besitzen eine eindeutigen ID, einen Namen und eine Rangnummer, die für die Reihung der Ebenen verwendet wird.

- **Informationen zu Orten:** Die Informationen die dem/der BenutzerIn zur Verfügung gestellt werden, können aus unterschiedlichen Quellen stammen. Die Daten zu einem gewählten Punkt können entweder direkt aus einem Webservice oder aus einem Pool von hinterlegten Daten kommen. Bei Webservices können unterschiedliche Services wie zum Beispiel Wikipedia oder Google Maps herangezogen werden, solange diese eine API zum Abrufen der Informationen zur Verfügung stellen. Zusätzlich können Daten direkt für bestimmte Karten in einer Datenbank hinterlegt werden. Diese werden dazu mit den Koordinaten und einer Ebene verknüpft hinterlegt. Das Auswählen der Informationen zu einem Punkt erfolgt durch das Sammeln von Orten in einem gewissen Umkreis (Abhängig des Zoomlevels) und das Zusammenfügen der Daten.

Alle diese oben genannten Punkte werden in einer relationalen Datenbank gespeichert, um diese Punkte global an einer Stelle sammeln und warten zu können. Diese Daten werden mittels einer Webschnittstelle aus der Datenbank gelesen. Externe Services werden in diesem Prototypen direkt von der Software aus angesprochen. Diese sollen bei einer finalen Version ebenfalls in einer Datenbank konfiguriert und somit dynamisch zugewiesen werden können.

#### Anpassungen zum Konzept von Moser

Im Vergleich zum Konzept von Moser wird in diesem Prototypen lediglich ein Datenpunkt mit Längen- und Breitengrad für jede Information in der Datenbank hinterlegt. Die Verwendung von Polygonen, um ein größeres Gebiet zu markieren, wird im Rahmen dieser Diplomarbeit nicht unterstützt.

## 5.2 Implementierung Software

Für die Implementierung der Software wurde die Entwicklungsumgebung Apple Xcode in der Version 7.2.1 und als Programmiersprache Swift 2 verwendet. Auf Grund der sehr umfangreichen und vielseitigen Entwicklung ist es im Rahmen dieser Diplomarbeit nur möglich Teile der Entwicklung vorzustellen. Der Fokus liegt auf der Vorstellung der wichtigsten Module und Codestellen, die sich als sehr schwierig herausgestellt haben. Des Weiteren wird versucht alle aufgetretenen Tücken, die im Rahmen der Entwicklung aufgetreten sind, vorzustellen. Der gesamte Sourcecode kann auf Github[3] eingesehen werden.

### 5.2.1 Aufbau der Software

Die Software besteht aus einem `ViewController`, der für das Handling der View inklusive aller Events zuständig ist. Diese Datei stellt das Herzstück der Software dar und beinhaltet alle wichtigen Funktionen für die Erkennung der unterschiedlichen Karten und Gesten. Beim Laden dieses Controllers werden alle notwendigen Definitionen und Konfigurationen der Variablen für die Software vorgenommen. Zusätzlich gibt es Klassen, die für die Verwaltung unterschiedlicher Objekte, wie zum Beispiel Zustände, Karten und individuelle Definitionen von Gesten, zuständig sind. Fehlende Informationen, die bei Bedarf für die Entwicklung notwendig waren, wurden aus der Apple Developer Library [Appb] sowie Büchern, über die Implementierung von Applikation in Swift 2 in iOS 9 wie von Nahavandipoor[Nah15] und Neuburg[Neu15], herangezogen.

Die Entwicklung der Software startet mit der Definition, der für die Applikation notwendigen Zustände und der Erkennung der benötigten Gesten. Durch die Verknüpfung dieser beiden Grundlagen kann das Interaktionskonzept abgedeckt werden. Im nächsten Schritt wird die Sprachausgabe implementiert und jeder Name der erkannten Geste als Audioinformation wiedergegeben. Anschließend folgt die Erstellung der Datenbank zur Verwaltung der Daten, sowie die Integration dieser in die Applikation. Zuletzt wird die automatische Kartenerkennung durch die Frontkamera implementiert. Die Vorstellung der folgenden Module erfolgt in dieser Reihenfolge der Entwicklung.

Um das Testen während der Implementierung zu erleichtern, wurde das Zustandsdiagramm visuell in der Applikation abgebildet. Dadurch lässt sich, sofern keine Karte auf dem Sensor liegt, der Ablauf der Zustände einfach verfolgen. Zusätzlich wird jeder Text, der über die Text-to-Speech Funktion ausgegeben wird, auch schriftlich am Display angezeigt. Beide dieser Punkte sind lediglich für das Testen des Prototypen gedacht und müssen in der Endausbaustufe nicht vorhanden sein.

### 5.2.2 Applikationszustände

Zum Speichern des aktuellen Zustandes wurde eine Variable `currentState` des Typs `State` im ViewController definiert, welche die ID des aktuellen Zustandes speichert. Um

[3] https://github.com/derro/MultiTouchPrototyp

den Key eindeutig zu gestalten, wurden alle möglichen Zustände in einem Enum mit einem String hinterlegt. Durch dieses Enum ist es möglich nur Inhalte dessen auszuwählen. Bei einem Wechsel der Zustände müssen mehrere Aufgaben erledigt werden. Aus diesem Grund wurde eine Funktion `func changeState(state:State)` definiert. Diese Funktion bekommt als Parameter den neuen Zustand des Typs `State` übergeben und legt diesen in der zuvor beschriebenen Variable `currentState` ab. Da in manchen Situationen bekannt sein muss, welcher Zustand zuvor aktiv war, wird auch dieser in einer Variable `lastState` gespeichert. Um Fehler bei der Gestenerkennung zu vermeiden, werden bei jedem Zustandswechsel alle Gesten-Recognizer aus der aktuellen View entfernt und nur jene zugewiesen, die für diesen Zustand relevant sind. Listing 5.1 zeigt einen Ausschnitt der Funktion `changeState()`, in der nur die benötigten Gesten-Recognizer für den Zustand *Layers* gesetzt werden.

```
func changeState(state :State) {
  disableAllGestureRecognizer()

  switch(state) {
    case State.Layers: stateLayers.backgroundColor = active;
      view.addGestureRecognizer(swipeLeft)
      view.addGestureRecognizer(swipeRight)
      view.addGestureRecognizer(swipeUp)
      view.addGestureRecognizer(swipeDown)
      view.addGestureRecognizer(doubleTap)
      view.addGestureRecognizer(rotationRecognizer)
      view.addGestureRecognizer(pinchRecognizer)
  ..
}
```

Listing 5.1: Codeausschnitt der Funktion `changeState()`

Für die Zustände *Hilfe, Einstellungen, Ebenen* und *Gespeicherte Orte* gibt es Subzustände. Diese sind ebenfalls, je Zustand, in einem Enum definiert. Die Definition ist notwendig um eine Navigation zwischen den Subzuständen zu ermöglichen.

### 5.2.3 Gesten

Dorau[Dor11] definiert eine Geste als *"eine Abfolge von Ereignissen, die jeweils charakteristisch für die Geste ist"*. Diese Charakteristika teilen sich laut Dorau in unterschiedliche Bereiche auf: Elementare Ereignisse, Verkettung, Bewegungsformen, Zeitsteuerung und Dynamik. Die Ausprägung dieser Bereiche ist für jede Geste unterschiedlich, wodurch sie sich eindeutig identifizieren lässt.

Alle im Konzept definierten Gesten mit Ausnahme des *Tap&Hold* und des Rotors konnten durch die von iOS zur Verfügung gestellten Recognizer abgedeckt werden. Diese zwei Spezialfälle werden in diesem Abschnitt später separat behandelt. Die von iOS fertigen

Funktionen zur Erkennung der Gesten sind konkrete Implementierungen der abstrakten Klasse `UIGestureRecognizer`, die den Subklassen eine Reihe an Funktionalitäten zur Verfügung stellt. Eine Vorstellung dieser wurde bereits im Kapitel 4.2.2 vorgestellt.

Bei der Arbeit mit Gesten-Recognizern ist die Unterscheidung zwischen diskreten und kontinuierlichen Gesten essentiell. Diskrete Gesten wie zum Beispiel ein *Double Tap* senden nur ein einziges Event, während kontinuierliche Gesten wie zum Beispiel ein *Swipe links* bei jeder Änderung der Touchpunkte ein Event schickt, bis die Geste abgeschlossen ist. Jeder Gesten-Recognizer arbeitet bei der Erkennung in einem endlichen Automat. Zu jedem Zeitpunkt befindet sich der Recognizer in einem bestimmten Zustand dessen. Abbildung 5.6 zeigt den Aufbau der Zustandsmaschine für die Gestenerkennung. Dieses Zustandsdiagramm ist wichtig für die Entwicklung, da bei zahlreichen Gesten, wie zum Beispiel *Pinch*, geprüft werden muss, in welchem Zustand sich die Geste befindet, bevor eine Aktion ausführt wird. Ohne einer Prüfung können Aktionen mehrfach oder zum falschen Zeitpunkt ausgelöst werden. In dieser Implementierung werden gemäß dem Transaktionsprinzip von Dorau[Dor11] nur Aktionen ausgeführt, wenn die Geste den Zustand *Recognized* erreicht hat. Ist dies nicht erfüllt oder die Geste abgebrochen wird (Zustand *Cancelled*), bleibt die Software im Zustand vor der Ausführung der Geste. Die Tabelle 5.2 zeigt eine Übersicht über alle von Apple iOS zur Verfügung gestellten Gesten-Recognizer und der Kategorisierung in diskrete und kontinuierliche Gesten.

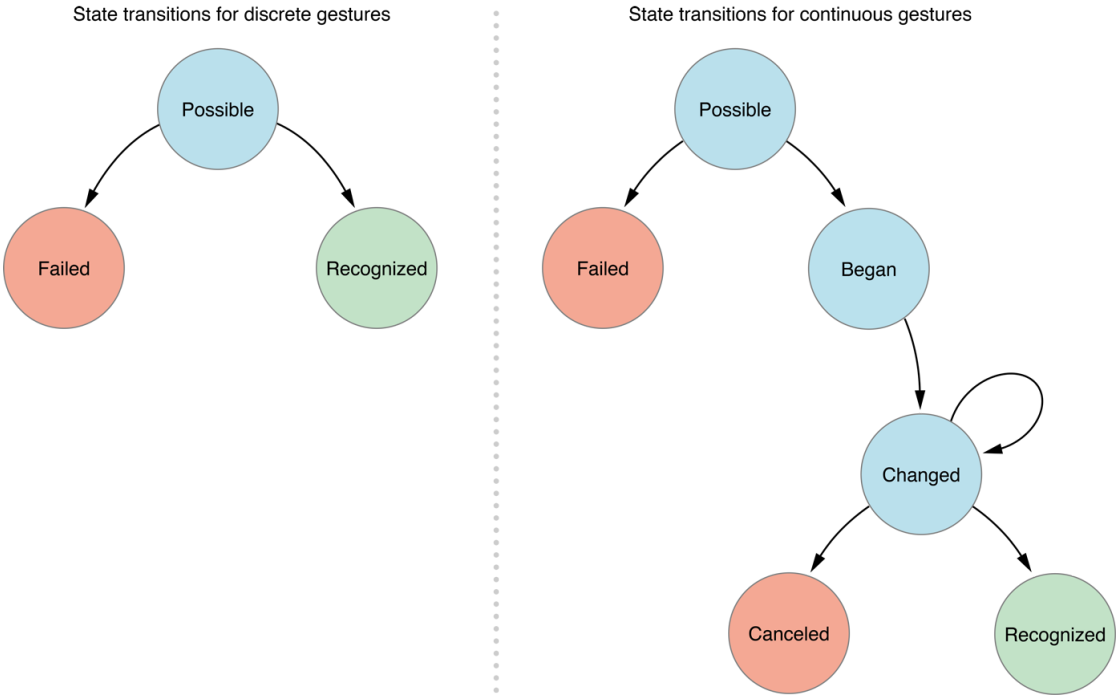


Abbildung 5.6: Zustandsdigaramm der Gesten-Recognizer von iOS[Appa] für diskrete (Abbildung links) und kontinuierliche Gesten (Abbildung rechts).

| Gesten Recognizer | Typ | Gesten |
|---|---|---|
| UITapGestureRecogizer | diskret | Tap, Double Tap(1), Double Tap(4), Driple Tap(4) |
| UIPinchGestureRecogizer | kontinuierlich | Pinch Open (2), Pinch Close (2) |
| UIRotationGestureRecogizer | kontinuierlich | Rotor |
| UISwipeGestureRecogizer | kontinuierlich | Swipe Links, Swipe Rechts, Swipe nach oben, Swipe nach unten |
| UIPanGestureRecogizer | kontinuierlich | - |
| UIScreenEdgePanGestureRecogizer | kontinuierlich | - |
| UILongPressGestureRecogizer | diskret | - |

Tabelle 5.2: Übersicht über alle von Apple iOS zur Verfügung gestellten Gesten-Recognizer, deren Typ, sowie die Gesten die für dieses Konzept darunter erkannt werden können.

Die Konfiguration der Gesten-Recognizer erfolgt beim Starten der Applikation. Dabei wird für jede zu erkennende Geste ein Recognizer definiert und konfiguriert. Je nach Typ können unterschiedliche Einstellungen wie Mindestanzahl der Touches, Mindestanzahl der Taps oder die Richtung gesetzt werden. Zusätzlich wird jedem Recognizer eine Zielfunktion angegeben, die beim Eintreten der Geste aufgerufen wird. Listing 5.2 zeigt ein Beispiel für die Konfiguration der Gesten *Double Tap* und *Swipe Links*.

```
// Konfiguration des Gesten Recognizers Tap
  let doubleTap = UITapGestureRecognizer()
  self.doubleTap.addTarget(self, action: "handleDoubleTap")
  self.doubleTap.numberOfTapsRequired = 2
  self.view.addGestureRecognizer(doubleTap)
  gestureRecognizers.insert(doubleTap)

// Konfiguration des Gesten Recognizers Swipe
  let swipeLeft = UISwipeGestureRecognizer()
  self.swipeLeft.direction = .Left
  self.swipeLeft.numberOfTouchesRequired = touchesForSwipe
  self.swipeLeft.addTarget(self, action: "handleSwipeLeft:")
  gestureRecognizers.insert(swipeLeft)
```

Listing 5.2: Konfiguration der Gesten-Recognizer der Gesten *Double Tap* und *Swipe Links*

Standardmäßig ist die Applikation so konfiguriert, dass immer nur eine Geste gleichzeitig ausgeführt und erkannt wird. Da man in diesem Konzept darauf angewiesen ist, dass mehrere Gesten parallel ausgeführt werden können (zum Beispiel im Zustand *Ort*: Tap & Hold und Swipe Links), muss dies separat konfiguriert werden. Dies kann durch

die Überlagerung der Funktion `func gestureRecognizer(gestureRecognizer: UIGestureRecognizer, shouldRecognizeSimultaneouslyWithGestureRecognizer otherGestureRecognizer: UIGestureRecognizer) -> Bool` mit dem Rückgabewert `true` erreicht werden.

### Tap&Hold

Wie bereits im Kapitel 4.3.2 aufgezeigt, wird für die Geste Tap & Hold eine individuelle Implementierung der Gestenerkennung implementiert. Hintergrund dafür ist, dass die Geste Tap & Hold mit dem von iOS zur Verfügung gestellten `UILongPressGestureRecognizer` nicht der gleichen Interpretation entspricht. Im Konzept ist vorgesehen, dass die benutzende Person die Karte mit den Fingern erkundet und bei Bedarf nach Informationen alle Finger bis auf jenen, der auf die gewünschten Stelle zeigt, vom Display nimmt. In der Interpretation dieser Geste von iOS darf kein Finger, nach dem Berühren des Touchscreens, bewegt werden. Bei einer Bewegung des Touchpunktes wird die Erkennung der Geste abgebrochen und nicht erkannt. Anzudenken ist, die in diesem Konzept verwendete Bezeichnung Tap & Hold auf Move & Hold umzuändern, um Verwechslungen zu vermeiden.

Die individuelle Implementierung der Erkennung von Tap & Hold basiert auf der Überlagerung der Methoden `touchesBegan`, `touchesMoved`, `touchesEnded` und `touchesCancelled`. Wird der Beginn dieser Geste erkannt, wird ein Countdown des Objektes `NSTimer` gestartet, der nach Ablauf der hinterlegten Zeit eine Funktion zur Verarbeitung der Geste aufruft. Wird die Methode `touchesMoved` aufgerufen, wird der Timer auf den ursprünglichen Wert zurückgesetzt, da kein halten der Position mehr vorliegt. Die weiteren zwei Methoden `touchesEnded` und `touchesCancelled` deaktivieren den Timer. Diese Methoden sind so konfiguriert, dass sie lediglich bei einer (Tap & Hold (1)) oder zwei (Tap & Hold (2)) Berührungen am Sensor im Zustand *Entdeckungsmodus* ausgeführt werden.

### Rotor

Die Komponente Rotor ist eine im Konzept definierte Geste, für die es keine, von iOS standardmäßig zur Verfügung gestellte, Methoden für die Erkennung gibt. Für die Realisierung dieses Konzeptes wird alternativ der `UIRotationGestureRecogizer` verwendet. Dieser Recognizer erkennt die Rotation von zwei gegenüberliegenden Touches in einer Kreisbewegung. Dabei werden die Werte der Rotation, Richtung und Geschwindigkeit analysiert. Diese Funktion zur Gestenerkennung wird dabei so verwendet, dass je nach Wert der Rotation ein Segment des Rotors ausgewählt wird. In den Phasen `UIGestureRecognizerState.Began` und `.Changed` der Erkennung wird der Name des Segmentes ausgegeben, sofern dieser sich zum vorigen ausgegebenen Wert unterscheidet. In der Phase `.Ended` wird der selektierte Zustand ausgewählt. Listing 5.3 zeigt die fertige Implementierung der Funktion für das Konzept des Rotors.

```
func handleRotation(gesture: UIRotationGestureRecognizer) {
```

```
// Wenn sich die Gestenerkennung im Zustand Began oder Changed
   befindet wird das Segment ausgegeben
  if(gesture.state == UIGestureRecognizerState.Changed ||
   gesture.state == UIGestureRecognizerState.Began)
  {
    if(gesture.rotation > 0.25 && gesture.rotation < 1.9 &&
   lastRotation != "Help") {
      speech.speak(getTextKey("Rotation_Help"));
      lastRotation = "Help"
    }
    if(gesture.rotation < -0.25 && gesture.rotation > -1.9 &&
   lastRotation != "Settings") {
      speech.speak(getTextKey("Rotation_Settings"));
      lastRotation = "Settings"
    }
    if(gesture.rotation < 0.25 && gesture.rotation > -0.25 &&
   lastRotation != "Back") {
      speech.speak(getTextKey("Rotation_Back"));
      lastRotation = "Back"
    }
  }

// Wenn sich die Gestenerkennung im Zustand Ended befindet,
   wird der Zustand ausgewaehlt
  if(gesture.state == UIGestureRecognizerState.Ended)
  {
    if(gesture.rotation > 0.25 && gesture.rotation < 1.9) {
      changeState(State.Help)
    }
    if(gesture.rotation < -0.25 && gesture.rotation > -1.9) {
      changeState(State.Settings)
    }
    lastRotation = ""
  }
}
```

Listing 5.3: Funktion für die Implementierung des Rotor Konzeptes.

### Abhängigkeiten zwischen Gesten

Die ersten Versuche haben gezeigt, dass die Erkennung der Gesten bei genauer Ausführung sehr zuverlässig erfolgt. Bei einigen Tests wurde festgestellt, dass die unsaubere oder schnelle Ausführung der Geste *Swipe* in manchen Situationen zu ei-

ner falschen Erkennung durch die Software führt. Anstelle der Geste *Swipe* wird dabei öfters die Geste *Pinch Close* oder *Pinch Open* erkannt. Dieses Verhalten konnte nur in der Kombination der Gesten Recognizer `UIPinchGestureRecognizer` und `UISwipeGestureRecognizer` nachgestellt werden. Aus diesem Grund wurde eine Rangordnung der Methoden zur Gestenerkennung vergeben. Durch die Verwendung der Funktion `func requireGestureRecognizerToFail(_ otherGestureRecognizer: UIGestureRecognizer)` kann eine Beziehung zwischen zweier Recognizer aufgebaut werden. Der Aufruf dieser Funktion auf dem Objekt des Pinch-Recognizers definiert, dass der übergebene Recognizer zuerst den Status *UIGestureRecognizerStateFailed* erreichen muss, bevor diese Geste in den nächsten Zustand gehen darf. Konkret wurde die Funktion wie folgt verwendet:

```
pinchRecognizer.requireGestureRecognizerToFail(swipeRight).
```

Diese Zeile Code definiert, dass das Objekt `pinchRecognizer` erst mit der Gestenerkennnung starten darf, wenn das Objekt `swipeRight` den Zustand *UIGestureRecognizerStateFailed* erreicht hat.

### 5.2.4 Interaktionskonzept

Nach der Definition der Applikationszustände und Gesten in den ersten Schritten der Implementierung, werden diese im Zuge des Interaktionskonzeptes miteinander verknüpft. Bei diesem Vorgang werden abhängig des aktuellen Zustandes und der ausgeführten Geste, Übergänge in andere Zustände ausgeführt. Die Übergange werden in Event Handlern der jeweiligen Gesten-Recognizer implementiert und bilden damit das im Konzept definierte Zustandsdiagramm (Abbildung 5.6) ab.

Der Aufbau ist für alle Event-Handler gleich gestaltet. Der aktuelle Zustand wird in einer `Switch`-Bedingung geprüft und die jeweilige Aktion für den Übergang (Funktion `changeState()`) oder Aktionen innerhalb des Zustandes ausgeführt. Die Funktion `changeState()` wurde bereits im Abschnitt 5.1.1 näher beschrieben.

### 5.2.5 Sprachausgabe

Für die Sprachausgabe wird die von iOS zur Verfügung gestellte Text-to-Speech Funktion der Klasse `AVSpeechSynthesizer` des Paketes `AVFoundation` verwendet. Diese Funktion steht seit der iOS Version 7.0 zur Verfügung und bietet Funktionen zur Ausgabe von Text, Steuerung und Monitoring der Sprache. Für die Ausgabe von Text wird ein Objekt `AVSpeechUtterance` erstellt und der auszugebende Text als Parameter übergeben. Dieses Objekt wird im Anschluss dem Synthesizer für die Ausgabe übergeben. Der Synthesizer speichert alle übergebenen Objekte in einer Liste (Queue) und arbeitet diese Objekte einzeln nacheinander ab. Findet während der Übergabe eines neuen Objektes gerade eine Ausgabe statt, wird diese dabei nicht unterbrochen und das neue Objekt am Ende der Liste angehängt.

Zur Steuerung bietet die Sprachausgabe Methoden für das Pausieren und Stoppen der Ausgabe. Während nach einer Pause an der gleichen Stelle fortgesetzt wird, endet bei Stopp die Ausgabe vollständig und alle vorhandenen Einträge der Warteschlange werden entfernt. Eine Funktion für das Überspringen eines Eintrages in der Warteschlange steht nicht zur Verfügung. Für das Monitoring ist ein Delegate Protokoll `AVSpeechSynthesizerDelegate` vorgesehen, dass Events während der Sprachausgabe sendet.

Die Implementierung im Prototyp erfolgt über eine Klasse `Speech`, die alle benötigten Funktionen für die Sprachausgabe kapselt. Bei der Initialisierung eines Objektes dieser Klasse werden alle Einstellungen wie die Lautstärke, Sprechgeschwindigkeit und Tonhöhe gesetzt. Dabei wird der Wert für die Sprechgeschwindigkeit aus dem globalen Parameter `AVSpeechUtteranceDefaultSpeechRate` entnommen, der in den zentralen Einstellungen des Betriebssystems gesetzt werden kann. Damit der/die BenutzerIn die Applikation nicht verlassen muss, kann die Geschwindigkeit der Sprache in den Einstellungen der Applikation verändert werden. Der Wert wird in den Usereinstellungen am Gerät gespeichert, damit dieser auch nach dem Beenden und neu Starten der Applikation vorhanden ist. In den Spracheinstellungen der Applikation kann der/die BenutzerIn durch die Geste *Swipe Links* die Geschwindigkeit um 10% verringern und durch die Geste *Swipe Rechts* um 10% erhöhen. Der von der Software zur Verfügung gestellte Bereich liegt zwischen 50% und 150%.

Die Klasse `Speech` bietet grundsätzlich drei unterschiedliche Methoden für die Ausgabe von Sprache:

- **speakText:** Diese Methode bekommt einen Text als Typ `String` übergeben und gibt diesen als Sprache aus. Läuft bereits eine Sprachausgabe, wird der neue Text in der Warteschlange hinzugefügt und, nachdem die vorigen Texte abgearbeitet sind, ausgegeben. Diese Funktion besitzt einen Parameter `interrupt` der beim zugewiesenen Wert `true` die aktuelle Ausgabe unterbricht, die Warteschlange leert und nur den neu übergebenen Text ausgibt. Standardmäßig ist der Parameter mit dem Wert `false` hinterlegt. Diese Methode wird für die Ausgabe von Namen der Zustände, Hilfetexte, Menüeinträgen und einzelnen Textausgaben verwendet.
- **speakArray:** In den Zuständen *Ort*, *Ort Details* und *Ebenen Details* werden mehrere Informationen hintereinander ausgegeben. Im Unterschied zur Variante `speakText` kann hier der/die BenutzerIn durch die Gesten *Swipe Links* und *Swipe Rechts* zwischen den Sprachausgaben vor- und zurückschalten. Der Funktion `speakArray` wird ein Array mit Texten übergeben und startet mit der Ausgabe der ersten hinterlegten Information. Durch die Funktionen `speakNext` kann die nächste und bei `speakPrev` die vorherige Information des Arrays ausgegeben werden. Diese Funktionen werden dabei entweder durch die Ausführung der *Swipe*-Gesten oder durch ein Event über das Beenden der vorigen Ausgabe aufgerufen. Für das Event wird dabei im `ViewController` eine Delegate-Funktion verwendet, die nach dem Beenden einer Sprachausgabe prüft, ob es sich um die Ausgabe

eines Array handelt und im positiven Fall den nächsten Text des Arrays ausgibt. Wird die Funktion `speakNext` oder `speakPrev` aufgerufen, wird die aktuelle Sprachausgabe abgebrochen und mit dem nächsten beziehungsweise vorigen Eintrag fortgefahren.

- **speakOrderdDictionary:** Diese Funktion verhält sich gleich zur Funktion `speakArray` mit dem einzigen Unterschied, dass die Datenstruktur der Datenquelle eine geordnetes Dictionary[4] ist anstelle eines Array. Da bei den Zuständen *Ebenen* und *Gespeicherte Orte* die Reihenfolge der Aktivierung beziehungsweise der Speicherung wichtig ist, wird diese Datenstruktur `OrderedDictionary` verwendet. Am Modus der Sprachausgabe ändert sich nichts.

Für die Funktionen `speakOrderdDictionary` und `speakArray` sind auch Varianten vorhanden, die einen Startindex als Parameter erwarten. Dieser Index gibt an, an welcher Stelle im Array oder dem Dictionary mit der Ausgabe begonnen werden soll. Diese Funktionen werden verwendet, wenn aus den Detail Zuständen von *Ebenen* oder *Gespeicherte Orte* in den Übersichtszustand zurück gewechselt wird, um bei gleicher Position in der Liste fortzufahren.

Im Listing 5.4 wird die Erzeugung des Synthesizers sowie dessen Standardeinstellungen gezeigt. Zusätzlich wird die Funktion `speak` als Beispiel der obigen Beschreibung angeführt. Alle diese Einträge sind aus der Klasse `Speech` entnommen.

```
// Erzeugung eines Synthesizer Objektes
let speechSynthesizer = AVSpeechSynthesizer()

// Einstellungen
var speechRate: Float = AVSpeechUtteranceDefaultSpeechRate
var speechPitchMultiplier: Float = 1.0
var speechVolume: Float = 1.0

// Funktion Speak zur Ausgabe einzelner Texte
func speak(text: String, interrupt: Bool) {
  // Falls der Parameter interrupt auf TRUE -> unterbreche die
   aktuelle Ausgabe
  if(interrupt) {
    stopSpeech()
  }

  // Erzeugung des Objektes speechUtterance
  let speechUtterance = AVSpeechUtterance(string: text)
```

[4] In einem geordneten Dictionary werden Wertepaare (Schlüssel/Id, Wert) geordnet in einer Liste abgelegt. Die Ordnung erfolgt dabei nach der Reihenfolge in der die Einträge der Liste hinzugefügt werden.

```
    // Setzen der konfigurierten Parameter
    speechUtterance.rate = speechRate
    speechUtterance.pitchMultiplier = speechPitchMultiplier
    speechUtterance.volume = speechVolume

    // Ausgabe des Textes
    speechSynthesizer.speakUtterance(speechUtterance)
}
```

Listing 5.4: Beispiel Funktion des Event Handlers der Geste Swipe nach unten

**Earcons**

Um dem/der BenutzerIn den Unterschied zwischen den Informationen beziehungsweise Ebenen zu erleichtern werden gezielt Earcons bei der Sprachausgabe mit einbezogen. Ziel ist es einen Ton pro Aktion zu definieren, welcher dem/der BenutzerIn bei dieser Aktion ausgegeben wird. Diese Töne sind nur bei der Ausgabe von Arrays oder Dictionaries in Verwendung.

Das Konzept sieht vor, dass Earcons in folgenden Situation ausgegeben werden:

- **Neue Information:** Bei der Auswahl eines Ortes wird eine Information zu jeder Ebene, falls vorhanden, hintereinander ausgegeben. Um eine Trennung zwischen den Ebenen wahrnehmen zu können, soll ein kurzer Ton den Beginn einer neuen Information ankündigen.
- **Ende einer Liste:** Wird das Ende einer Liste erreicht, soll ein Ton dem/der BenutzerIn mitteilen, dass keine weiteren Ausgaben mehr erfolgen.
- **Anfang einer Liste:** Wird der Anfang einer Liste erreicht, teilt ein Ton dem/der BenutzerIn mit, dass ein weiter zurückblättern nicht möglich ist und der Start erreicht wurde.

Um dem/der BenutzerIn eine eindeutige Interpretation zu ermöglichen ist es wichtig, dass für jede Bedeutung ein eigener Ton verwendet wird. Abbildung 5.7 zeigt eine grafische Darstellung einer Sprachausgabe mit mehreren Informationen und Tönen. Die Sprachausgabe startet dabei immer mit dem Earcon der ersten Information. Bei dem Wechsel der Ebene wird vor der Ausgabe der Sprache ein weiterer Ton ausgegeben. Wird das Ende erreicht wird dem/der BenutzerIn ebenfalls ein Ton abgespielt.

Bei der Implementierung wurde auch mit der Verwendung von Systemtöne des Betriebssystems iOS experimentiert. Auf Grund der Verwendung der Kamera und des automatischen Shutter Geräusches, dass bei der Erstellung eines Fotos automatisch abgespielt wird, wurden diese aber anschließend in den Systemeinstellungen deaktiviert um den/die BenutzerIn nicht zu verwirren. Ein einzelnes Deaktivieren des Geräusches war im Zuge

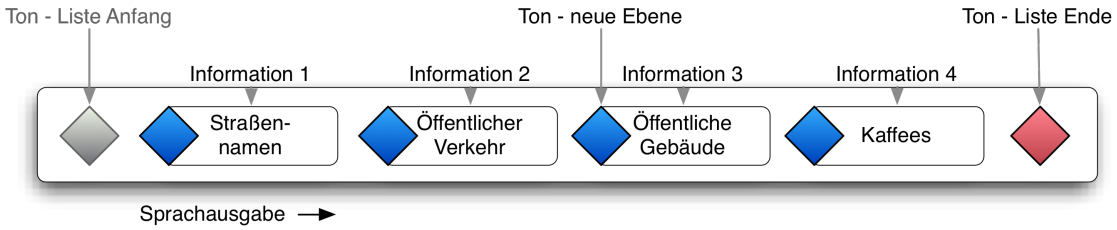


Abbildung 5.7: Grafische Darstellung des Ablaufes der Sprachausgabe. Vor jeder neuen Information (Wechsel der Informations-Ebene), sowie beim Erreichen des Endes der Liste wird dem/der BenutzerIn ein Ton ausgegeben. Die Sprachausgabe startet mit dem ersten Ton der ersten Information. Der Earcon am Anfang der Liste wird nur ausgegeben, wenn der User bis an den Anfang zurückblättert.

der Implementierung leider nicht möglich. Um trotz der deaktivierten Systemtöne Earcons abspielen zu können wurde für die Ausgabe das Software Modul `AVAudioPlayer` verwendet. Diesem Modul werden die Töne als MP3-Datei zum Abspielen übergeben.

### Textbausteine

Um die Wartbarkeit der Applikation zu erhöhen, werden alle Texte zur Sprachausgabe in einer Property Liste gespeichert. Eine Property Liste ist ein strukturiertes Text File, das bei der Entwicklung von iOS Applikationen für die Speicherung von Werten im Schlüssel/Wert (Key/Value) - Format verwendet wird. Die Struktur wird durch die Verwendung von `XML` vorgegeben.

Alle Texte werden dort als Key/Value-Paar eingetragen und können in der Software bei Bedarf über den Key ausgelesen werden. Die Werte sind als Typ `String` definiert.

Listing 5.5 zeigt einen Auszug der Property Liste der Textbausteine TextKeys.plist.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://
   www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    // Definition der Zustands Textbausteine
    <key>State_Help</key><string>Hilfe </string>
    <key>State_Settings</key><string>Einstellungen </string>
    <key>State_Discover</key><string>Entdeckungsmodus </string>
    <key>State_Location</key><string>Ort </string>
    <key>State_Layers</key><string>Ebenen </string>
    <key>State_CustomLocation</key><string>Eigene Orte </string
   >
```

```
<key>State_CustomLocationSelected</key><string>Ort
selektiert </string>
<key>State_Distance</key><string>Distanz </string>
<key>State_Settings_Speech</key><string>Spracheinstellungen
</string>
<key>State_Settings_Layers</key><string>Ebenen
Einstellungen </string>
<key>State_Settings_SavedLocation</key><string>Gespeichete
Orte </string>
..

// Definition der Gesten Textbausteine
<key>Gesture_doubleTap</key><string>Doppeltes tippen </
string>
<key>Gesture_pinchOpen</key><string>Zwei Finger auseinander
ziehen </string>
<key>Gesture_pinchClose</key><string>Zwei Finger zusammen
ziehen </string>
..
```

Listing 5.5: Defintion der Textbausteine in der TextKeys.plist Property Datei.

### 5.2.6 Kartenerkennung

Für die Kartenerkennnug wird, wie bereits im Konzept angeführt, ein optischer Ansatz verfolgt. Dazu werden alle haptischen Karten mit einem farbigen Papier ausgestattet, dass nach der Platzierung der Karte auf den Multi-Touch Sensor die Kamera überdeckt. Die Applikation erstellt in regelmäßigen Abständen Fotos mit der Frontkamera und wertet die Farbwerte dieser Bilder aus. Die erkannte Farbe wird mit den konfigurierten Karten abgeglichen und bei einem Treffer wird diese Karte als aktiv hinterlegt und in den Zustand *Entdeckungsmodus* gewechselt. Das Intervall der Fotos ist in der Software konfigurierbar und wurde nach den ersten Tests auf einen Wert von 2 Sekunden gesetzt. Wurde eine Karte erfolgreich erkannt, erhöht sich das Intervall auf 10 Sekunden um Ressourcen des Gerätes zu sparen. Beim Entfernen der Karte wird die Applikation in den Zustand *Start* zurückgesetzt und das Intervall auf 2 Sekunden reduziert.

Listing 5.6 zeigt die Initialisierung der Frontkamera und des Timers für die automatische Kartenerkennung.

```
// Wenn die automatische Erkennug aktiviert ist
if(cameraSensor == true) {
    captureSession.sessionPreset = AVCaptureSessionPresetLow
    // Auslesen aller Devices
    let devices = AVCaptureDevice.devices()
    for device in devices {
```

```
        // Test ob dieses Device Video unterstuetzt
        if (device.hasMediaType(AVMediaTypeVideo)) {
            // Test ob es sich dabei um die Frontkamera handelt
            if(device.position == AVCaptureDevicePosition.Front
   ) {
                captureDevice = device as? AVCaptureDevice
            }
        }
    }

    // Initialisierung des Timers
    cameraTimer = NSTimer.scheduledTimerWithTimeInterval(
   fireCameraTime, target: self, selector: "fireMapCheck:",
   userInfo: nil, repeats: true)
}
```

Listing 5.6: Codeausschnitt zur Initialisierung der Frontkamera des iPad Pro, sowie des Timers

Für Testzwecke wurde ein Parameter `cameraSensor` des Typs `Boolean` hinterlegt um die automatische Kartenerkennung deaktivieren zu können. Im deaktivierten Modus kann zwischen dem Zustand *Start* und *Entdeckungsmodus* mit Hilfe der Geste *Double Tap (1)* gewechselt werden. Als Karte wird dabei immer eine vordefinierte Karte geladen, ohne dass diese auf dem Multi-Touch Sensor aufliegen muss.

Um die Farben aus dem Bild zu extrahieren, sowie die Farbwerte mit der Karte zu vergleichen, wurden drei Extensions entwickelt und einzelne Objekte damit um Funktionen erweitert.

- Die erste Extension erweitert das Objekt `UIImage` um eine Funktion `getPixelColor(pos: CGPoint)`. Diese Funktion, die auf jeden Bild des Typs `UIImage` aufgerufen werden kann, bekommt eine Position im Bild als Parameter übergeben. Die Funktion liefert die Farbwerte dieses Punktes als Typ `UIColor` retour.

- Die zweite Extension `getColorDifference(fromColor: UIColor)` erweitert das Objekt `UIColor`. Als Parameter wird ein Farbwert des Typs `UIColor` übergeben und die Differenz mit dem Objekt, für das die Funktion aufgerufen wurde, verglichen. Der Rückgabewert ist die Differenz beider Farbwerte.

- Die letzte Extension `getNextMatch()` erweitert ebenfalls das Objekt `UIColor`. Diese Funktion vergleicht die Farbe mit den hinterlegten Farbwerten der Karten und gibt den besten Match mit der geringsten Differenz aus, sofern ein gewisser Grenzwert unterschritten ist. Wird dieser Wert überschritten, liegt kein Treffer vor und es wurde keine passende Karte gefunden.

Die Erkennung der Farbe funktioniert nach folgendem Prinzip: Jedes Foto, das über die Frontkamera des iPad Pro erstellt wird, wird anhand der darin vorkommenden Farben analysiert. Dazu werden in jeweils fünf Segmente, an unterschiedlichen Stellen des Fotos, die Farbwerte aller Pixel zu einem Farbwert zusammen gemischt um Ungleichmäßigkeiten im Foto auszugleichen. Die Ergebnis der fünf Segmente werden dabei im Anschluss wieder zusammengemischt, so dass ein einzelner Farbwert als Repräsentant für das Foto zur Verfügung steht. Dieser Farbwert wird mit den Referenz-Farbwerten der Karten abgeglichen und die Differenz daraus berechnet. Der Wert mit der niedrigsten Differenz (sofern dieser unter einem gewissen Grenzwert liegt) stellt das Ergebnis der Kartenerkennnung dar. Diese Art von Erkennung ist auf Grund der Einfachheit ein äußert gebräuchlicher Modus und ist in zahlreichen Applikation in dieser Art und Weise vertreten.

### 5.2.7 Datenkonzept

Da mehrere Applikationen gleichzeitig auf unterschiedlichen Sensoren laufen können, wie zum Beispiel während eines Unterrichts in einer Schulklasse, sollen die Daten zentral in einer Datenbank abgelegt sein. Durch diese zentrale Ablage kann jedes Gerät auf die gleichen Daten zugreifen. Änderungen an der Datenbasis können damit zentral an einem Ort vorgenommen werden, ohne den Code oder Dateien der Applikationen zu aktualisieren.

Um den Aufwand des Backends zu reduzieren wird auf ein bereits verfügbares Online Backend namens *Parse*[5] zurückgegriffen. Parse bietet eine breite Palette an Funktionen wie eine Online Datenbank, Server Side Code, Logging, Monitoring und vieles mehr. Zusätzlich bietet Parse zahlreiche vorgefertigte Anbindungen für Applikationen in unterschiedlichen Programmiersprachen. Dabei werden Frameworks für die Entwicklung von iOS Applikationen zur Verfügung gestellt. Diese bieten fertige Funktionalitäten für die Authentifizierung und den Abruf der Daten aus der Online Datenbank. Für die Anforderungen im Rahmen dieses Prototypen ist es ausreichend die Online Datenbank zu verwenden, sowie die zur Verfügung gestellten Frameworks in iOS. Die Kosten von Parse sind abhängig von der Anzahl an verwendeten Services und Usern. Für die Implementierung eines Prototypen im Rahmen dieser Diplomarbeit ist die von Parse zur Verfügung gestellte kostenlose Variante ausreichend. Diese Version bietet bis zu 30 Anfragen pro Sekunde - einen Wert, der für die Zwecke eines Prototypen ausreichend ist.

Alle für den Prototyp relevanten Daten werden in der Datenbank in drei Tabellen abgelegt. Diese drei Tabellen sind wie folgt aufgebaut:

- **Map:** Die Tabelle Map (Karte) speichert alle notwendige Informationen der konfigurierten Karten.
  - `objectId:String` - eindeutige ID des Datenbankeintrages
  - `name:String` - Bezeichnung der Karte

[5] http://www.parse.com

  - `info:String` - Infotext über die Karte, der beim Auflegen dieser dem/der BenutzerIn ausgegeben wird.
  - `coordinates:Array` - Array mit allen Koordinaten der Karteneckpunkte. Diese Werte dienen zur Berechnung der Touchpunkte auf der Karte.

- **Layers:** Die Tabelle Layers (Ebenen) beinhaltet alle konfigurierten Ebenen und deren Eigenschaften.
  - `objectId:String` - eindeutige ID des Datenbankeintrages
  - `key:String` - eindeutiger Key für das Mapping der Textbausteine
  - `name:String` - Bezeichnung der Ebene
  - `active:Boolean` - Flag zum Markieren, ob diese Ebene aktiv ist und somit in der Software angezeigt werden soll.
  - `defaultOrder:Number` - Standardreihung der Ebenen für die Ausgabe der Information in der Applikation.

- **MapData:** Die Tabelle MapData (Kartendaten) beinhaltet alle zu den Karten hinterlegten Informationen. Die Inhalte dieser Tabelle werden sowohl mit Einträgen der Tabelle Map als auch Layers verknüpft.
  - `objectId:String` - eindeutige ID des Datenbankeintrages
  - `message:String` - Infotext zum aktuellen Ort und Ebene
  - `active:Boolean` - Flag zum Markieren, ob diese Information aktiv ist und somit in der Software ausgegeben werden soll.
  - `map:Pointer<Map>` - Fremdschlüssel zu einem Eintrag der Tabelle Map.
  - `map:Pointer<Layer>` - Fremdschlüssel zu einem Eintrag der Tabelle Layer.
  - `location:GeoPoint` - Koordinaten des Punktes, zu dem diese Information gehört. Die Angabe erfolgt in Breiten- und Längengrad.

Abbildung 5.8 zeigt die grafische Oberfläche der Datenbank und Beispielinhalte der Tabelle MapData.

Nach der Fertigstellung der Entwicklung kündigte Parse ein Ende des Dienstes an. Dieser Dienst wird noch bis zum 28. Jänner 2017 angeboten und danach eingestellt. Als Alternative wird ein Open Source Parse Server angeboten um sich ein eigenes Parse Backend zu bauen. Weiters kann auch auf ähnliche Projekte wie zum Beispiel Firebase[6] zurückgegriffen werden.

[6] https://www.firebase.com

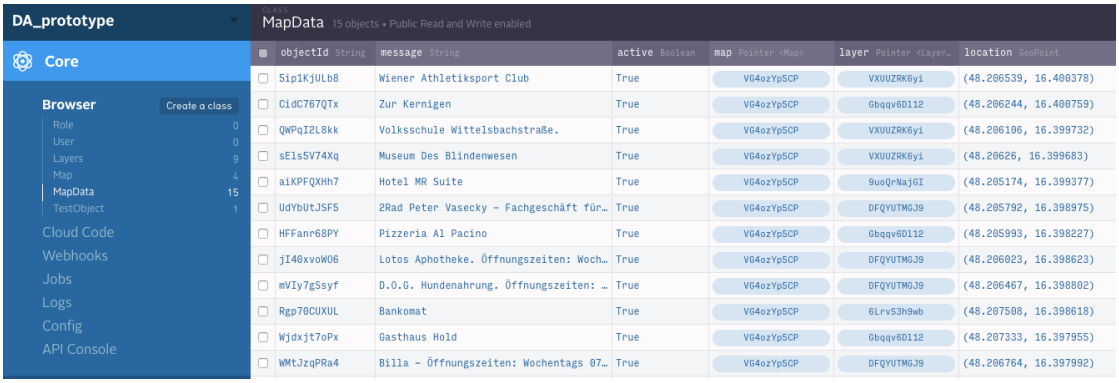


Abbildung 5.8: Übersicht über die Inhalte der Tabelle MapData in der Datenbank.

### Verarbeitung der Daten

Um das Framework Parse in Xcode einzubinden, muss lediglich die SDK von Parse heruntergeladen und der Applikation hinzugefügt werden. Nach dem Hinzufügen der Bibliotheken im iOS Projekt, steht Parse für die Benutzung zur Verfügung. Für den Abruf von Daten aus der Datenbank stellt Parse eine eigene API zur Verfügung [Par]. Mit Hilfe von `Queries` können unterschiedliche Abfragen an der Datenbank durchgeführt werden. Die Daten selbst werden anschließen als Objekt des Typs `PFObject` empfangen. Jedes `PFObject` stellt Schlüssel/Werte-Paare mit JSON kompatiblen Daten dar. Zusätzlich zu Standardabfragen können auch datenabhängige Abfragen erstellt werden. Für den Typ `GeoPoint` können zum Beispiel Klauseln wie `whereKey:nearGeoPoint` oder `whereKey:nearGeoPoint:withinKilometers` angegeben werden. Die Klausel `whereKey:nearGeoPoint` filtert dabei nach Einträgen, die Nahe der übergebenen Koordinaten liegen. Die Klausel `whereKey:nearGeoPoint:withinKilometers` liefert alle Einträge, die im Umkreis innerhalb eines bestimmten Wertes liegen. Diese Abfrage wird in dieser Applikation für die Sammlung der Daten zu einem Touch Punkt des Users verwendet. Listing 5.7 zeigt die Abfrage der Koordinaten und Verarbeitung der Ergebnisse einer solchen Query.

```
func getPOI() {
  do {
    // Definieren des Touchpunktes
    let point = PFGeoPoint(latitude:self.lat, longitude:self.
   lng)

    // Definition der Abfrage auf die Tabelle MapData
    let query = PFQuery(className:"MapData")

    // Verlinkung der Tabelle Layer
    query.includeKey("layer")
```

```
    // Klausel - alle Punkte innerhalb von 100 Meter zum
   angegeben Punkt
    query.whereKey("location", nearGeoPoint:point,
   withinKilometers:0.01)

    // Absenden der Anfrage
    let allPOIs = try query.findObjects()

    // Iteration ueber alle Ergebnisse
    for obj in allPOIs {
      // Speichern der Informationen, falls der/die BenutzerIn
   diese Ebene aktiviert hat
      if((obj["layer"]["active"] as! Bool) == true) {
          information[(obj["layer"]["key"] as! String)] = (obj
   ["message"] as! String)
      }
    }
  } catch _ {
    // Fehlerfall - Errrorhandling
    puts("Error: getPOI()")
  }
}
```

Listing 5.7: Abfrage der Karteninformationen zu einem vom User gewählten Punkt auf der Karte.

#### Anbindung weiterer Services

In der Endausbaustufe des Konzept ist es Vorgesehen externe WebServices als Datenquelle anzubinden. Diese Services sollen die Grundlage der Daten für die Ausgabe der Informationen bieten. Um dieses Konzept im Zuge des Prototypen zu testen, wird die Google API als externes Service angebunden. Diese API wird verwendet, um die von dem/der BenutzerIn gewählte Koordinaten in einen Straßennamen umzuwandeln und auszugeben. Zusätzlich wird die Google API für die Berechnung der Distanz zwischen zwei Punkten verwendet. Die Verwendung der Google API ist abhängig der Anfragen pro Tag/Sekunde gratis oder kostenpflichtig. Das Limit mit 2500 Abfragen pro Tag und 10 Abfragen pro Sekunde ist für diese Implementierung ausreichend und somit kostenlos. Für die Endausbaustufe sollen andere freie Services angedacht werden.

Listing 5.8 zeigt die Abfrage der Straßennamen über die Google API.

```
func getStreetName(completion: (result: String) -> Void) {
  // Definition der API URL inklusive Koordinaten als Parameter
```

```
    let query = "https://maps.googleapis.com/maps/api/geocode/
     json?latlng=\(lat),\(lng)&location_type=ROOFTOP&result_type=
     street_address&key=<KEY>"

    // Umwandlung des Query-Strings in eine NSURL
    guard let url = NSURL(string: query) else {
      print("Error - getStreetName(): cannot create URL")
      return
    }
    let urlRequest = NSURLRequest(URL: url)

    // Verarbeitung des asynchronen Requests
    let task = NSURLSession.sharedSession().dataTaskWithRequest(
     urlRequest, completionHandler: { (data, response, error) in
      let json = JSON(data: data!)
      // Auslesen des Namens
      let streetname = "\(json["results"][0]["address_components
     "][1]["short_name"]) \(json["results"][0]["
     address_components"][0]["short_name"])"
      // Speichern der Daten
      self.information["streetnames"] = streetname
      // Notify ueber den Erhalt der Daten
      completion(result: streetname);
    })
    task.resume()
}
```

Listing 5.8: Abfrage des Straßennamen zu einem vom User gewählten Punkt auf der Karte.

### Ablauf der Datensammlung

Bei einer Auswahl eines Punktes auf der Karte durch den/die BenutzerIn, wird dieser Punkt von X,Y-Koordinaten am Multi-Touch Sensor von der Software in Länge- und Breitengrad umgerechnet. Mit Hilfe dieser zwei Werte werden alle Informationen aus den unterschiedlichen Datenquellen gesammelt. Dazu werden Informationen aus der Datenbank über die Parse API und zusätzlich der Straßenname mit Hilfe der Google API geladen. Diese Informationen werden zusammen in der Applikation in einem Array abgelegt und dem/der BenutzerIn als Information über die Sprachausgabe wiedergegeben. Dabei werden nur Informationen ausgegeben, deren Ebenen in den Ebenen Einstellungen aktiviert sind. Der/die BenutzerIn selbst merkt keinen Unterschied zwischen den verschiedenen Datenquellen aus denen die Informationen stammen.

Die Abbildung 5.9 zeigt eine Darstellung von möglichen Hotspots auf einer Karte mit den in der Datenbank hinterlegten Werten.

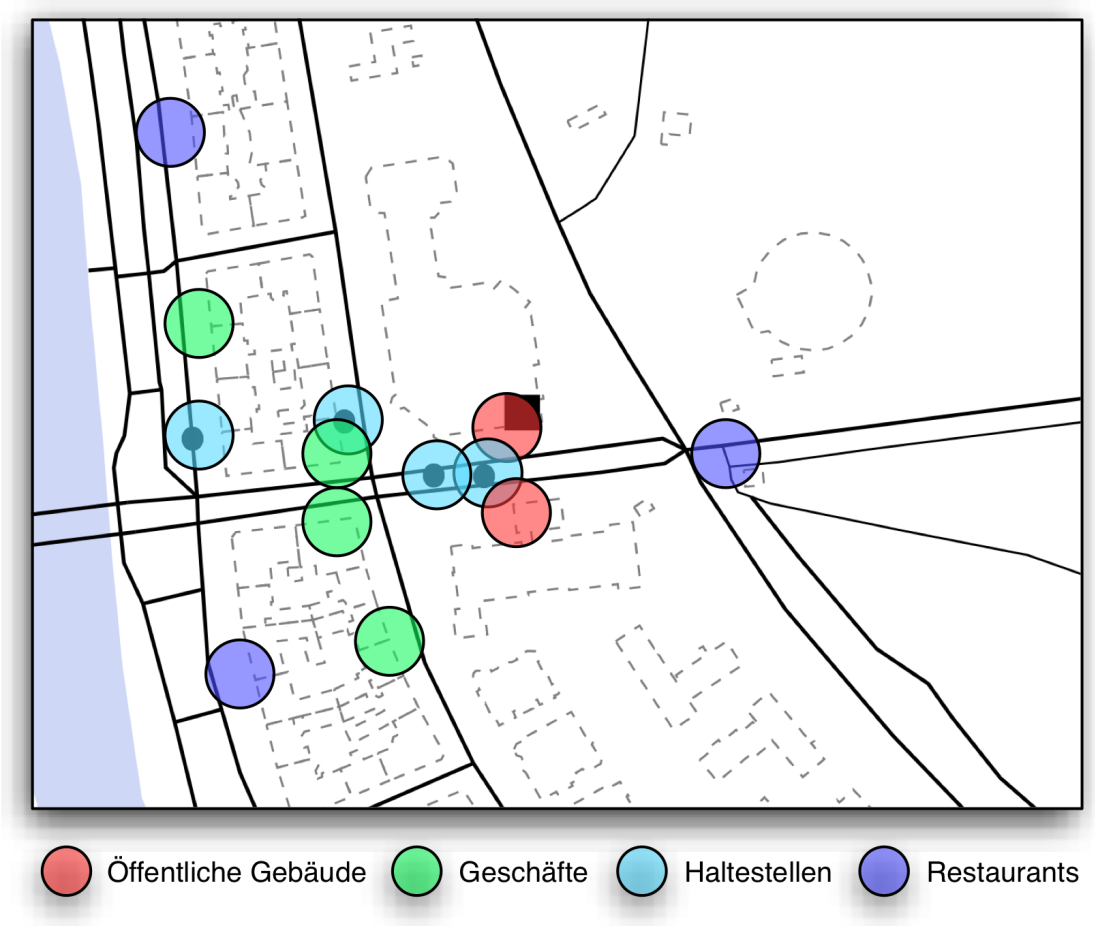


Abbildung 5.9: Darstellung der Information-Hotspots auf einer taktilen Karte.

## 5.3 Implementierung der Hardware

Dieser Abschnitt beschäftigt sich mit der Fertigstellung aller Komponenten, die nicht in die Kategorie Software fallen, aber für einen vollständigen Prototypen benötigt werden.

### 5.3.1 iPad Pro Hülle

Wie sich bereits bei den ersten Tests, welche im Kapitel 4.3.4 beschrieben sind, herausgestellt hat, wird eine Hülle für das iPad Pro benötigt. Diese ist notwendig, um die haptischen Karten mit dem iPad Pro zu verbinden damit sich diese beim Benutzen nicht

verschieben. Bei einer falschen Ausrichtung kann es zu fehlerhaften Informationen der selektierten Punkte kommen.

Zusammengefasst ergeben sich folgende Anforderungen:

1. Die haptischen Karten müssen sich schnell und einfach wechseln lassen.
2. Ein Verrutschen der Karten darf nicht möglich sein. Eine optimale Ausrichtung der Karten an den Sensor muss gegeben sein.
3. Karten im Format DIN-A4 müssen unterstützt werden.
4. Eine Handhabung mit der Hülle muss auch durch sehbehinderte Menschen möglich sein. Es darf keine Verletzungsgefahr bestehen.
5. Das Gehäuse muss entweder fest am iPad Pro verbaut werden, oder wenn es modular ist, schnell und einfach anbringen lassen. Im Idealfall kann dieses Zusammenbauen auch durch eine blinde Person durchgeführt werden.
6. Die Anschlüsse des iPad Pro müssen zur Interaktion frei bleiben. Zusätzlich darf die Frontkamera nicht verdeckt sein.
7. Die Tasten zur Regulierung der Lautstärke müssen zugänglich sein.
8. Das iPad Pro darf durch das Gehäuse nicht beschädigt werden.
9. Der Interaktionsraum (der Bereich über der Karte) muss für die Interaktion mit den Händen frei bleiben.

Dank der Zusammenarbeit mit Professor Purgathofer und dem Institut für Gestaltungs- und Wirkungsforschung konnte eine Reihe an Möglichkeiten evaluiert werden. Neben den Materialien die man in normalen Schreibwarengeschäften und Baumärkten erwerben kann, standen so zusätzlich die Möglichkeiten eines 3D-Druckers und eines Laser Cutters zur Verfügung.

Die Website `thingiverse.com` bietet zahlreiche Projekte, die mit einem 3D Drucker oder Laser Cutter erstellt werden können. Dabei werden die Beiträge durch BenutzerInnen der Community hochgeladen und der Öffentlichkeit zur Verfügung gestellt. Eines dieser Projekte namens “iPad Pro Case“[Thi] vom 19.12.2015 konnte zahlreiche der oben definierten Anforderungen abdecken. Das Projekte beinhaltet eine Vorlage für das Ausschneiden mehrerer Einzelteile durch einen Laser Cutter, die später zu einer iPad Pro Hülle zusammengebaut werden. Diese Hülle war in diesem Projekt zur Montage an der Wand gedacht.

Zur vollständigen Erfüllung aller Anforderungen musste die Vorlage noch leicht modifiziert werden. Die Hülle war so designend, dass lediglich der Sensor des iPad Pro frei lag. Dieser Bereich musste vergrößert werden, um ein Blatt der Größe DIN-A4 auf den Sensor legen

zu können. Im Vergleich zum Projekt auf *Thingiverse* wurde für die Hülle des Prototypes, anstelle von Plexiglas, Holz mit einer Dicke von 4mm verwendet. Die geschnittenen Einzelteile können der Grafik 5.10 entnommen werden.

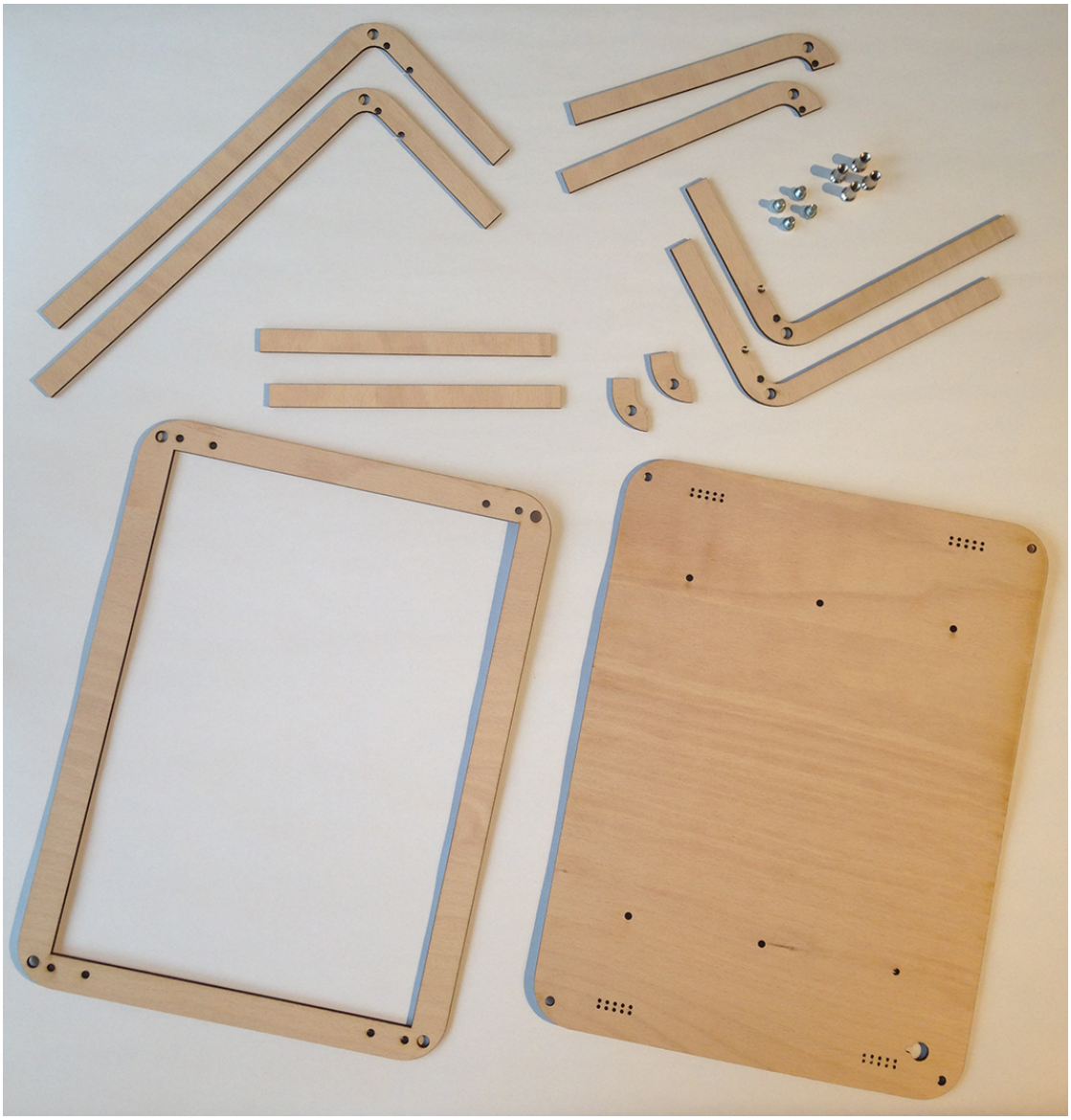

Abbildung 5.10: Geschnittene Einzelteile des LaserCutter für die iPad Pro Hülle.

Die Lautstärke Regler des iPad Pro's sind nach dem Verbau im Gehäuse auf Grund des breiten Rahmens nicht einsetzbar. Um die Lautstärkeregler zu verwenden wurde mittels eines 3D-Druckers ein Einsatz zur Verlängerung der Tasten an den Rand des Gehäuses erstellt. Der Taster ist in der Abbildung 5.11 zu sehen.

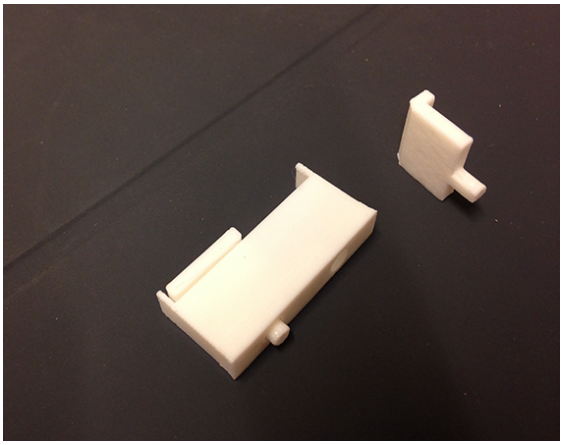

Abbildung 5.11: Aus dem 3D-Drucker entworfene Bauteile zur Verlängerung des Lautstärkereglers auf die Außenseite der iPad Pro Hülle.

Das Zusammenbauen der Einzelteile erfolgte mit Leim und Schrauben. Die Oberseite der Hülle ist dabei nicht verleimt, damit diese zum Einsetzen des iPad Pro's heruntergenommen werden kann. Die Innenseite des Gehäuse wurde mit Moosgummi ausgelegt um einen besseren Halt zu bekommen und eine Beschädigung des iPad Pro's zu verhindern. Abbildung 5.12 zeigt die fertige Hülle.

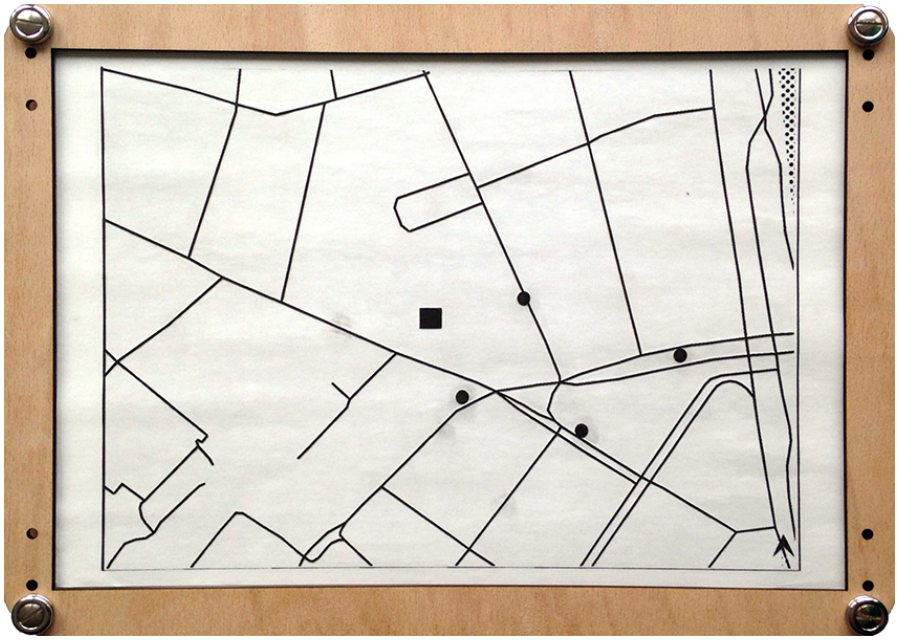

Abbildung 5.12: Fertiggestellte iPad Pro Hülle mit dem Beispiel einer taktilen Karte.

### 5.3.2 Haptische Karten

Im Rahmen des Prototypen wurden zwei vollständige taktile Karten entworfen, die bei der Evaluierung der Implementierung den blinden Personen zur Verfügung stehen. Für die Auswahl des Bereiches der ersten Karte, wurde das Bundes-Blindenerziehungsinstitut (BBI) in Wien[7] als Zentrum genommen. Das Zoom-Level ist dabei so eingestellt, dass die Karte in etwa einen Bereich von 400 x 300 Meter abdeckt. Dies entspricht in etwa der Abbildung von drei bis fünf Querstraßen. Die zweite Karte wurde direkt links an die erste angelegt um das zu erkundende Gebiet zu vergrößern. Die Abbildung 5.13 zeigt dabei die Auswahl der Kartenausschnitte, sowie das BBI Wien als Zentrum. Der Bereich der ersten Karte ist dabei mit einem blau gefüllten Rechteck und jener der zweiten Karte mit einem grün gefüllten Rechteck markiert.

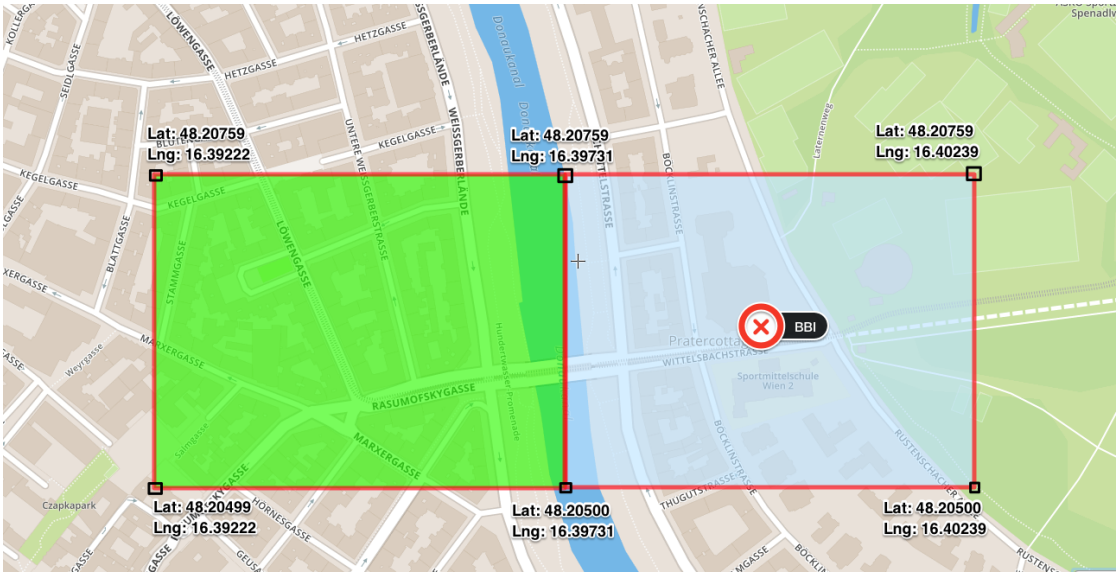


Abbildung 5.13: Auswahl der zwei Kartenausschnitte für die Erstellung der haptischen Karten. Das Bundes-Blindenerziehungsinstitut ist in der rechten Karte (blaues Rechteck) mit einem roten X markiert.

Nach der Definition der Bereiche wurden diese mit Hilfe einer speziell entwickelten Software für den Druck auf einem Braille-Drucker optimiert. Die Software wandelt dabei alle Straßen in Linien um und Gebäudeumrandungen werden strichliert auf der Karte dargestellt. Öffentliche Haltestellen werden für das leichtere Auffinden mit einem schwarzen Viereck markiert. Für die später richtige Ausrichtung durch den/die BenutzerIn wird an der rechten unteren Ecke noch eine Kompassnadel zur Orientierung angebracht. Abbildung 5.14 zeigt die Kartenausschnitte einer Karte vor und nach dem Umwandeln durch die Software. Die Software wurde von Florian Holzner im Rahmen seiner Bachelorarbeit an der TU Wien entwickelt.

[7] Adresse des Bundes-Blindenerziehungsinstitut: Wittelsbachstraße 5, 1020 Wien, Österreich


Abbildung 5.14: Ausschnitt der Karte aus Open Street Map und nach dem Umwandeln durch die Software CEAT. Links das Original aus Open Street Maps, rechts das fertige Bild für den Druck auf dem Braille-Drucker.

Nach der Umwandlung der Kartenausschnitte in für den Druck optimierte Bilder, werden diese Bilder pro A4 Blatt so ausgerichtet, dass die Karte genau den Bereich des iPad Pro Displays abdeckt. Durch diese Methode kann sicher gestellt werden, dass der/die BenutzerIn für jeden Touchpunkt auf der haptischen Karte eine Information durch die Applikation bekommt. Touchpunkte außerhalb des Sensors wären für die Applikation nicht erkennbar. Im nächsten Schritt werden die Karten auf einem Braille Drucker gedruckt, oder auf Schwellpapier kopiert.

Der Bereich, der anschließend die Frontkamera des iPad Pro überdeckt wird aus der haptischen Karte geschnitten und mit farbigen 80g Papier ersetzt. Diese Farben sind für die optische Kartenerkennung durch die Software notwendig. Anschließend ist noch die Konfiguration der Karte in der Datenbank notwendig. Dazu werden die Koordinaten der Eckpunkte eingetragen, sowie der Farbcode in der Applikation für diese Karte hinterlegt.

## 5.4 Zusammenfassung

Dieses Kapitel beschäftigt sich mit der Erstellung eines Konzeptes basierend auf den vorangegangen Diplomarbeiten. Dabei werden die Grundlagen aus den Diplomarbeiten von Sommersguter[Som13] und Moser[Mos12] entnommen, um neue Ideen und Konzepte erweitert, beziehungsweise der gewählten Technologie angepasst. Parallel dazu wurde ein Prototyp basierend auf diesem Konzept entwickelt. Erfahrungen und Testergebnisse die im Rahmen der ersten Tests gewonnen wurden nachträglich im Konzept verarbeitet und angepasst. Somit stellt das in diesem Kapitel vorgestellte Konzept (5.1) den vollständigen theoretischen Aufbau der Implementierung dar.

Der Prototyp selbst besteht aus einer iOS Applikation die auf einem iPad Pro ausgeführt

wird. Die komplette Applikationslogik befindet sich auf dem Multi-Touch Sensor und die auszugebenden Daten werden aus einer Datenbank oder externen Webservices entnommen. Der Abschnitt 5.2 beschreibt dabei, wie bei der Entwicklung vorgegangen wurde und welche Problematiken sich bei der Entwicklung ergeben haben.

Neben der Software und dem Konzept wurden auch die haptischen Karten und eine Hülle für das iPad Pro erstellt. Diese zwei Komponenten sind notwendig, um eine Evaluierung des Konzeptes von Anfang bis zum Ende durchführen zu können. Die Herstellung, sowie einige Bilder zur Veranschaulichung wurden im Abschnitt 5.3 vorgestellt.

KAPITEL 6

# Evaluation

Im Anschluss an die Entwicklung wurde eine Evaluierung des entstandenen Prototypen, durch die Unterstützung einer blinden Expertin des Braille-Zentrums Wiens, durchgeführt. Ziel der Evaluation war es, den erstellten Prototypen und dessen Soft- und Hardwarekomponenten auf Praxistauglichkeit zu prüfen. Frau Eva Papst, Leiterin des Braille-Zentrums, beschäftigt sich mit vielen unterschiedlichen Bereichen der Technik und deren Einsatz als alltagstaugliche Hilfsmittel für blinde Personen[1]. Bereits in den vorangegangen Arbeiten und Projekten in diesem Bereich wurde Frau Papst mehrmals als Expertin für Evaluierungen, Tests und Feedbackgespräche herangezogen.

## 6.1 Ablauf der Evaluierung

Die Evaluierung wurde auf dem Apple iPad Pro mit der eigens dafür entwickelten Software durchgeführt. Zusätzlich war das Gerät in ein speziell für diesen Zweck gebautes Gehäuse verbaut. Als taktile Karten wurden die im Kapitel 5.3.2 erstellten Karten verwendet. Die darauf abgebildeten POIs[2] wurden in der Datenbank mit den notwendigen Informationen hinterlegt.

Zu Beginn wurde der Hintergrund des Projektes kurz präsentiert, um den Fokus der Evaluierung auf die vorgesehen Ziele zu legen. Im Rahmen einer kurzen Vorstellung und Beschreibung konnten die einzelnen Komponenten des Prototypen der Versuchsperson näher gebracht werden. Durch das Ertasten wurde eine gewisse Vertrautheit mit dem Gerät hergestellt. Mit der Einstellung des Sprachtempos wurden die Vorbereitungen abgeschlossen und der erste Praxistest gestartet.

[1] Nähere Informationen zu Frau Eva Papst und ihren Gedanken zum Thema Blindheit sind auf ihrer persönlichen Webseite http://aus-meiner-feder.at zu finden.

[2] POI: Point of Interests, auf Deutsch übersetzt: Punkt von besonderem Interesse

Um eine Überforderung der Expertin zu vermeiden, wurde das Vorgehen einer sukzessiven Erklärung des Interaktionskonzeptes und der Gesten gewählt. Dabei wurde mit dem Kernelement, dem Erkundungsmodus, begonnen. In den weiteren Schritten wurden zusätzliche Funktionen des Konzeptes punktuell erklärt.

Zunächst wurde die Karte mit dem Zentrum des BBI auf den Sensor gelegt und die Software in den Zustand *Erkundungsmodus* versetzt. Die Ausgabe der Erstinformation über die Karte wurde durch die Software initiiert. Nach der Erklärung des Erkundungsmodus wurde Frau Papst mit der ersten Aufgabe, der Erkundung der Karte beauftragt. Die ersten Beobachtungen zeigten, dass die taktile Karte wesentlich vorsichtiger und nur mit wenigen Fingern erkundet wurde. Vergleichen hierzu wurde die Karte zu Beginn, ohne am Sensor zu liegen, mit allen zehn Fingern erkundet.

Um sich ein besseres Bild über das Gebiet zu verschaffen, wurden die haptischen Eindrücke durch die Ausgabe von geobasierten Informationen erweitert. Frau Papst wurde beauftragt sich die Straßennamen zu einzelnen Punkten auf der Karte ausgeben zu lassen. Im Zuge der Ausgabe der Informationen wurde das Konzept hinter den Tönen (Earcons) näher beschrieben und die weiteren Gesten zum Navigieren zwischen Informationen oder zum Speichern eines Ortes erläutert.

Nach der Besprechung der Gesten zum Wechseln zwischen den Zuständen, wurde eine neue Software auf das iPad geladen, welche die Geste *Pinch Open* im Zustand Entwicklungsmodus deaktiviert. Für die Erkundung der Straßen hat Frau Papst zwei Finger auf diese gelegt und sie durch das Verfolgen der Straßenverläufe in beide Richtungen ertastet. Das Auseinanderziehen der Finger wurde mehrmals als Geste *Pinch Open* interpretiert. Dies hatte zur Folge, dass die Erkundung durch den Wechsel in einen anderen Zustand erschwert beziehungsweise unterbrochen wurde. Eine sinnvolle Erkundung konnte damit nicht durchgeführt werden. Mit der Deaktivierung der Geste konnte die Erkundung ohne unbeabsichtigte Unterbrechungen fortgesetzt werden.

Im nächsten Schritt wurde die Funktionsweise für die Messung von Entfernungen erklärt und die Distanz zwischen mehreren Punkten testweise ermittelt. Diese Aufgabe konnte ohne Schwierigkeiten absolviert werden.

Abgeschlossen wurde der Erkundungsmodus durch das Abspeichern von Orten und der wiederholten Ausgabe der Informationen im Zustand *Gespeicherte Orte*. Für das Abspeichern der Orte sind zwei Hände notwendig um die vorgesehen Gesten auszuführen. Ein Finger bleibt auf dem selektierten Punkt um diesen im Fokus zu behalten, während die zweite Hand eine *Swipe*-Geste zum Speichern ausführt. Dabei wurde beobachtet, dass der Einsatz einer zweiten Hand für die Interaktion ungewohnt ist. Dies lässt sich durch die Tatsache begründen, dass die häufig verwendeten Smartphones im Vergleich zum iPad Pro kleiner sind. Deshalb wurde versucht, die beiden Gesten mit einer Hand hintereinander auszuführen.

Vervollständigt wurde die Evaluierung durch eine zweite Erkundungsrunde auf einer weiteren Karte. Im Unterschied zur ersten Karte war die Umgebung der Expertin weniger bekannt. Zusätzlich bildet sie ein dichter bebautes Gebiet mit vielen Kreuzungen und

parallelen Straßen ab. Es galt das Gebiet auf dieser Karte zu erkunden, sowie die Öffentlichen Haltestellen aufzufinden. Ziel dieser Runde war es zu sehen, ob die zuvor erlernten Aktionen ohne Hilfe wieder ausgeführt werden können.

Abbildung 6.1 zeigt Frau Papst bei der Erkundung der ersten taktilen Karte.

Abbildung 6.1: Frau Papst bei der Evaluierung des Prototypen auf Praxistauglichkeit.

## 6.2 Verarbeitung der Informationen

Bei der Evaluierung konnten sowohl zahlreiche gut umgesetzte Punkte, als auch Problemstellen im Konzept aufgezeigt werden. Im Rahmen der Nachbearbeitung wurde die

Audioaufzeichnung der Evaluierung mehrmals analysiert und alle identifizierten Punkte in einer Minde Map nach Themen gruppiert zusammengefasst. Im ersten Durchgang wurden die einzelnen Punkte notiert und grob den Themenblöcken *Ablauf*, *Verbesserungsvorschläge*, *Erkundungsmodus* und *Haptische Karte* zugeordnet. Zusätzliche Durchläufe führten zu einer verfeinerten und erweiterten Darstellung der gewonnen Informationen. Die Ergebnisse sind in der Mind Map in der Abbildung 6.2 angeführt und werden im folgenden Abschnitt näher beschrieben.

## 6.3 Erkenntnisse

Die durch die Evaluierung gewonnen Erkenntnisse reichen vom Einsatz der Hardwarekomponenten, bis hin zu Aktionen und Interpretationen in der Software. In diesem Abschnitt werden die wichtigsten gesammelten Punkte pro Kategorie näher präsentiert. Dabei wird auf einzelne Problemstellen und eventuelle Lösungsansätze eingegangen. Die Gliederung der Punkte entspricht der gleichen Kategorisierung wie in den Abschnitten 5.1 und 5.2 im Kapitel *Implementierung*.

### 6.3.1 Applikationszustände

Bei der Evaluierung lag der Fokus hauptsächlich auf den Zuständen *Erkundungsmodus*, *Ort*, *Gespeicherte Orte* und *Distanz*, da diese den Hauptteil der Funktionalität abdecken und dem/der AnwenderIn den gewünschten Mehrwert und positiven Lernerfolg bringen.

Erkenntnisse zum Thema Applikationszustände:

- **Erkundungsmodus:** Im Erkundungsmodus bleibt ein Finger auf einem Punkt stehen um dazugehörige Informationen zu erhalten. Bei der Evaluation konnte beobachtet werden, dass dieser Modus das erste Mal sehr gut funktioniert. Im Anschluss wird oftmals direkt weiter erkundet, ohne den Finger zu heben und den Zustand *Ort* zu verlassen, um in den Erkundungsmodus zurückzukehren. Dadurch ergibt sich der Nachteil, dass bei einem neuen gewählten Punkt keine Informationen ausgegeben werden und der/die BenutzerIn verwirrt ist. Als Lösungsansatz kann ein neuer Algorithmus entwickelt werden, welcher durch ein Bewegen des Fingers ebenfalls wie beim Heben des Fingers den Modus beendet.

- **Ort:** Im Zustand Ort werden die hinterlegten Informationen zu einem Punkt dem/der BenutzerIn ausgegeben, solange der Finger auf diesem Punkt ruht. Weitere Gesten zum Vor- und Zurückschalten zwischen Informationen oder zum Speichern von Orten müssen mit Hilfe der zweiten Hand ausgeführt werden. Es konnte bei der Evaluation festgestellt werden, dass der Einsatz einer zweiten Hand für Frau Papst sehr ungewohnt war und diese Gesten oftmals mit jener Hand ausgeführt wurde, die den Punkt zuvor fixierte. Dieses Verhalten wurde nur am Beginn festgestellt und hat sich im Laufe der Evaluierung gelegt.

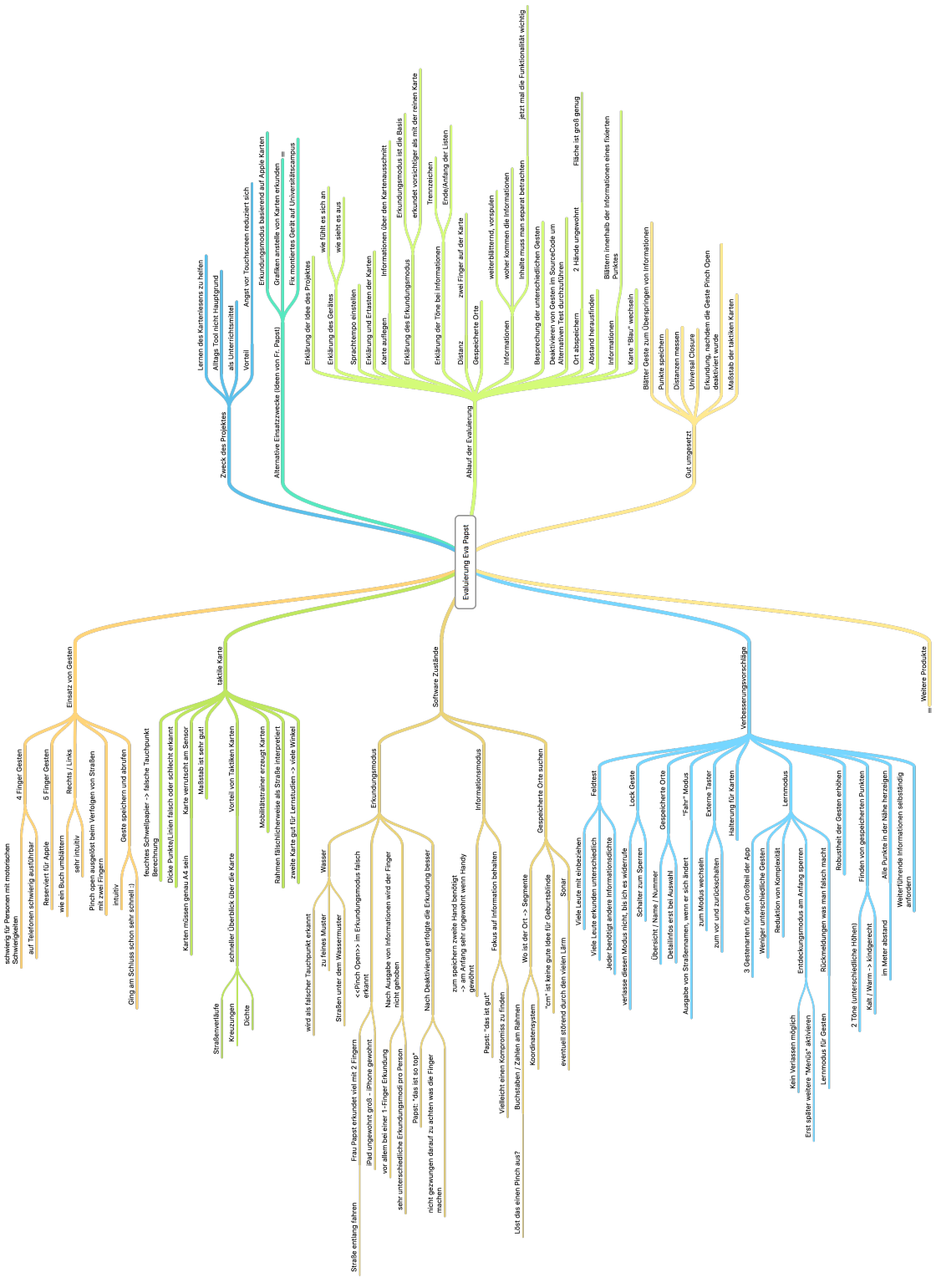


Abbildung 6.2: Gesammelte Erkenntnisse aus der Experten Evaluierung des Prototypen mit Frau Eva Papst in einer Mind Map.

### 6.3.2 Gesten

Die Gesten stellen das Kernelement der Interaktion zwischen dem/der BenutzerIn und dem Tablet dar. Aus diesem Grund sind die folgenden Erkenntnisse als sehr wichtig anzusehen und sollten unbedingt für einen reibungslosen Einsatz iterativ angepasst werden:

- **Vier Finger Gesten:** Der Einsatz von vier Fingern ist für die Geste *Swipe* und für die Ausgabe des aktuellen Zustandes vorgesehen. Diese Gesten können für Personen mit motorischen Schwierigkeiten schwer auszuführen sein. Durch die Anpassung der Fingeranzahl könnte diese Geste vereinfacht werden.
- **Pinch Open Geste:** Im Zuge der Erkundung mit zwei Fingern wurde mehrmals die Geste *Pinch Open* unabsichtlich ausgeführt. Vor allem bei der Erkundung von Straßen, bei denen die Finger vom Mittelpunkt der Straße jeweils ans Ende der Straße fahren, kann diese Geste fälschlicher erkannt werden. Dieses Problem könnte durch die Wahl einer alternativen Geste unterbunden werden.
- **Robustheit:** Die Robustheit der Gesten muss erhöht werden, damit diese nicht fälschlicherweise ausgeführt werden. Dies betrifft vor allem die Gesten *Pinch Open* und *Rotor*.

Positive Anmerkungen zu den definierten Gesten:

- **Intuitiv:** Die Geste *Swipe* wurde als sehr intuitiv für den eingesetzten Zweck, dem Vor- und Zurückblättern, bezeichnet. Dieses Konzept lässt sich in zahlreichen Applikationen wieder finden. Auch die Gesten zum Speichern von Orten (zum Körper ziehen) und Ansehen der gespeicherten Orte (vom Körper weg) wurde als sehr leicht zu merken und umzusetzen beschrieben.

### 6.3.3 Sprachausgabe

Zum Thema Sprachausgabe konnten keine Mängel im Zuge der Evaluierung festgestellt werden. Positiv angemerkt wurde die Möglichkeit, die Sprachgeschwindigkeit in den Einstellungen der Applikation zu konfigurieren.

### 6.3.4 iPad Pro Gehäuse

Durch das iPad Pro Gehäuse wird dem/der BenutzerIn die Möglichkeit gegeben, die zu erkundende Karte einfach und richtig auf dem Sensor zu platzieren.

Folgende Punkte konnten als Schwierigkeiten identifiziert werden:

- **Fixierung der Karten:** In dem Gehäuse wurde eine Fläche von der Größe eines A4-Blattes für die Positionierung der Karten freigelassen. Die zur Verfügung gestellten

Schwellkarten waren dabei um ein paar Millimeter größer, was zu einem Verrutschen der Karte während der Erkundung führte. Anzudenken ist ein Mechanismus, um die Karte mit dem Gehäuse fest zu verbinden oder die Karten exakt auf die benötigte Größe DIN A4 zuzuschneiden. Dabei ist es wichtig die exakte Ausrichtung des Kartenausschnittes mit dem Sensor des iPad Pro zu gewährleisten, um eine richtige Berechnung der Touchpunkte und Koordinaten zu erreichen.

Positive Anmerkungen zum iPad Pro Gehäuse:

- **Verarbeitung:** Die Verarbeitung des Gehäuse wurde beim Ertasten als sehr angenehm empfunden. Alle Ecken und Schrauben sind abgerundet und es können keinerlei Verletzungen entstehen.

### 6.3.5 Taktile Karten

Taktile Karten bieten blinden BenutzerInnen die Möglichkeit sich rasch Informationen über das dargestellte Gebiet zu besorgen. Im Zuge des Erstkontaktes mit der Karte, konnte Frau Papst durch das Ertasten der Karte wertvolle Informationen über die Dichte des bebauten Gebietes, die Anordnung der Straßen und Flüsse binnen weniger Sekunden ermitteln.

Erkannte Schwierigkeiten im Umgang mit taktilen Karten:

- **Wasser:** Flüsse wurden zur Vereinfachung für den/die BenutzerIn auf der taktilen Karte als ein feines Muster dargestellt. Hierdurch konnte der Sensor den Touchpunkt nicht immer eindeutig zuordnen, da aufgrund des zu feines Musters zahlreiche Touchpunkte auf dem Sensor entstanden sind. So wurde zum Beispiel ein Touchpunkt durch den Sensor als vier Touchpunkte interpretiert. Zusätzlich ist eine Erkennung von Brücken über einen Fluss durch die Überlagerung von Muster und den Straßenkonturen nur sehr schwer möglich. Als Gegenmaßnahme kann ein gröberes Muster für die Visualisierung der Wasserflächen verwendet werden.

- **Rahmen:** Um den vom Sensor erkannten Bereich zu markieren wurde ein Rahmen als Abgrenzung gelegt. Dieser Rahmen wurde bei der Evaluation mehrmals fälschlicherweise als Straße interpretiert. Eventuell ist hier eine Schraffierung des Rahmens anzudenken, um ihn von einer Straße eindeutig unterscheiden zu können.

Positive Anmerkungen zu den taktilen Karten:

- **Maßstab:** Der Maßstab der zur Verfügung gestellten Karten (ca. 400m x 300m) wurde für die Erkundung als sehr positiv angesehen.

## 6.4 Zusammenfassung

Im Rahmen der Evaluierung wurde mit Frau Eva Papst der entwickelte Prototyp einem ersten Praxistest unterzogen. Dabei wurde mit Hilfe zweier taktiler Karten das Konzept und die Idee des Gerätes vorgestellt und die Umsetzung evaluiert. Die gewonnen Erkenntnisse erstrecken sich, ausgehend von der Definition der Gesten, über die Funktionen der Zustände bis hin zum Einsatz der taktilen Karten. Zahlreiche Punkte des Konzeptes wurden als sehr nützlich und gut realisiert bewertet. Einzig kleine, sich wiederholende Fehlererkennung bei den Gesten führten zu teilweiser Verwunderung über die dadurch folgenden Aktionen. Die Evaluierung mit Frau Eva Papst war sehr wertvoll, um die Aktionen und Vorgänge von blinden Personen bei der Erkundung von taktilen Karten besser zu verstehen. Zusätzlich konnten zahlreiche Fragen über die Interpretation von Gesten und ihre Bedeutung für blinde Personen geklärt werden.

KAPITEL 7

# Fazit und Ausblick

Blinde Personen werden im Laufe ihres Lebens mit zahlreiche Barrieren konfrontiert. Dieses Projekt hat das Potential bei der Überwindung einer dieser Hürden, dem Erlernen von räumlichem Denken und dem Lesen von Landkarten, zu unterstützen. Durch die aktuellen, am Markt erhältlichen Technologien können Konzepte wie diese umgesetzt werden und zum Abbau von Barrieren beitragen.

## 7.1 Fazit

Das Ziel dieses Projektes war es, blinden Personen die Möglichkeit zu geben Umgebungen und Bereiche von Stadtkarten mittels audiounterstützter Software zu erkunden. Durch die Vermittlung von kartografischer Information und der Verknüpfung von bereits vorhandenem Umgebungswissen soll so das räumliche Denken aktiv unterstützt werden.

Im Rahmen der Analyse der Multi-Touch Technologien hat sich das iPad Pro als klarer Favorit für den Einsatz als Prototyp herausgestellt. Abschließend betrachtet lieferte dieses Tablet alle gewünschten Funktionen für die technische Implementierung der benötigten Funktionen. Kleinere Stolpersteine bei der Entwicklung der Software, wie zum Beispiel Limitierungen bei dem Einsatz von Gesten durch den Hersteller, konnten durch alternative Implementierungen und Anpassungen beseitigt werden. Fehlende Hardwareunterstützung, wie beispielsweise bei der automatischen Kartenerkennung durch RFID-Chips, konnte durch optische Ansätze mit Hilfe der verbauten Frontkamera gelöst werden.

Die taktilen Karten stellen die Basis der Informationen für den/die AnwenderIn dar. Alle Informationen der Karten werden auf Straßen, öffentlichen Haltestellen und Flüsse reduziert, um ein klar ertastbares Bild zu ermöglichen. Der Einsatz der Karten hat sich als sehr anwenderfreundlich herausgestellt und konnte, unter dem Einsatz von haptischen Informationen, rasch und effizient grobe Informationen über die Struktur und Beschaffenheit des Gebietes liefern. Einzig bei der Verwendung von Mustern für die

Abbildung von Flüssen muss die Erstellung der Karte zum besseren Erkunden angepasst werden. Das Wechseln der Karte ermöglicht unterschiedliche Gebiete abzudecken. Die Größe derselben Karte muss mit der freien Fläche des Tablet-Gehäuses übereinstimmen. Hier gibt es weiteres Optimierungspotential, das im Zuge der Evaluierung ermittelt wurde.

Das Interaktionskonzept zwischen dem/der BenutzerIn zeigt das meiste Potential für zusätzliche Verbesserungen. Einzelne Gesten führten zu Konflikten im Erkundungsmodus, da durch die Erkundung mittels unterschiedlicher Anzahl an Fingern, teilweise unbeabsichtigt Gesten ausgeführt werden. Die Anzahl an unterschiedlichen Möglichkeiten stellt gerade zu Beginn einer Anwendung ein kompliziertes Bild der Applikation dar und führt zu teilweisen Überforderung.

Die Anbindung an externe Services als Informationsquelle für Straßennamen hat sich als sehr zuverlässig erwiesen. Auch die statischen Daten in der Datenbank mit Informationen zu Restaurants, Geschäften und vielem mehr lieferte ein gutes Maß an Informationsdichte.

Das entstandene Konzept deckt das Umfeld für die Entwicklung des Projektes ab. Die Ansätze haben sich als sehr brauchbar und wertvoll im Rahmen der Evaluierung erwiesen und zeigen, dass es in die damit geplante Richtung geht. In der Evaluierung durch Frau Papst konnten einige noch vorhandene Schwierigkeiten, aber auch zahlreiche gut umgesetzte Punkte erkannt werden. Durch weitere Durchläufe von Implementierungsanpassungen und Feldtests mit einer größeren Anzahl an ProbandInnen könnten die noch vorhandenen Schwachstellen sukzessive beseitigt werden.

Das Resultat ist sehr zufriedenstellend und ein guter Grundstein, um dieses Gerät im Zuge der nächsten Schritte zu finalisieren. In weiterer Folge soll der Einsatz in Schulklassen von Blindenschulen im Rahmen des Mobilitätsunterrichts ermöglicht werden.

## 7.2 Ausblick

Für die endgültige Verwendung des Gerätes im Rahmen des Unterrichts gilt es weitere Verbesserungen umzusetzen, um einen erheblichen Mehrwert für den/die EndanwenderIn zu ermöglichen.

Die Verwendung des iPad Pro hat sich als sehr zuverlässig erwiesen. Auf Grund der hohen Anschaffungskosten sollten alternative Technologien für den Einsatz evaluiert werden. Im Laufe der Diplomarbeit wurden neue Geräte am Verbrauchermarkt vorgestellt, die ähnliche Funktionsweisen und Leistungen aufweisen. Eine Verwendung von Technologien, die haptische Informationen ausgeben können, würde das aufwendige Erstellen der taktilen Karten entfallen, zum Beispiel: Braille-Display für eine flexible Darstellung.

Die momentane Nutzung von statisch gespeicherten Daten könnte durch die Anbindung von Webservices abgelöst werden. Dafür ist ein Mechanismus zur Konfiguration und Verwaltung dieser Dienste nötig, sowie eine sorgsame Validierung der zur Verfügung gestellten Daten.

Die Verwendung des Konzeptes sollte nicht nur auf den hier ausgewählten Anwendungsfall reduzierten werden. Alternative Einsatzzwecke sind ebenfalls denkbar. Anstelle von taktilen Karten können im Schulkontext eingesetzte Grafiken, wie zum Beispiel das menschliche Skelett, auf Schwellpapier abgebildet und erkundet werden. Auch ist ein Einsatz als fest verbaute Übersichtskarte eines Universitätsgeländes möglich.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Listings

# Literaturverzeichnis


[And] Andreas Hegenberg, Leap Motion, Inc. Leap Motion App Store - BetterTouch Tool. https://apps.leapmotion.com/apps/bettertouchtool/osx. zuletzt besucht am: 19.1.2016.

[Appa] Apple Inc. Event Handling Guide for iOS - Gesture Recognizers. https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/GestureRecognizer_basics/GestureRecognizer_basics.html. zuletzt besucht am: 20.03.2016.

[Appb] Apple Inc. iOS Developer Library. https://developer.apple.com/library/prerelease/ios/navigation/. zuletzt besucht am: 20.03.2016.

[Appc] Apple Inc. iPad Pro - Technische Daten. http://www.apple.com/de/ipad-pro/specs/. zuletzt besucht am: 31.03.2016.

[Appd] Apple Inc. VoiceOver Getting Started - Using VoiceOver Gestures. https://www.apple.com/voiceover/info/guide/_1137.html. zuletzt besucht am: 18.03.2016.

[Arm13] Armin Wagner, Georg Kaindl - Institut für Design und Gestaltungswissenschaften der Technischen Universität Wien. Collaboratively Editable Audio Tangibles (CEAT). https://igw.tuwien.ac.at/ceat, 2013. zuletzt besucht am: 19.1.2016.

[Arm14] Armin Wagner, Georg Kaindl - Institut für Design und Gestaltungswissenschaften der Technischen Universität Wien. An Open Capacitive Multi-Touch Tracker. http://www.wiretouch.net, 2014. zuletzt besucht am: 19.1.2016.

[Bil14] Bill Buxton - Microsoft Research. Multi-Touch Systems that I Have Known and Loved. http://www.billbuxton.com/multitouchOverview.html, 2014. zuletzt besucht am: 29.3.2016.

[Buna] Bundesministerium für Wissenschaft, Forschung und Wirtschaft. Sparkling Science - Sparkling Fingers. http://www.sparklingscience.at/de/projects/show.html?--typo3_neos_nodetypes-page[id]=8. zuletzt besucht am: 03.03.2016.

[Bunb] Bundesministerium für Wissenschaft, Forschung und Wirtschaft. Sparkling Science - Sparkling Fingers 2.0. http://www.sparklingscience.at/de/projects/show.html?--typo3_neos_nodetypes-page[id]=410. zuletzt besucht am: 06.03.2016.

[CCL13] Dustin Carroll, Suranjan Chakraborty, and Jonathan Lazar. Designing Accessible Visualizations: The Case of Designing a Weather Map for Blind Users. In *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion*, pages 436–445. Springer Berlin Heidelberg, Berlin, Heidelberg, July 2013.

[CFRV15] Giorgio Colombo, Giancarlo Facoetti, Caterina Rizzi, and Andrea Vitali. Low Cost Hand-Tracking Devices to Design Customized Medical Devices. In *Universal Access in Human-Computer Interaction. Access to Learning, Health and Well-Being*, pages 351–360. Springer International Publishing, Cham, July 2015.

[Dor11] Rainer Dorau. *Emotionales Interaktionsdesign.* Gesten und Mimik interaktiver Systeme. Springer-Verlag, Berlin, Heidelberg, February 2011.

[EK08] Florian Echtler and Gudrun Klinker. touch software architecture. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, NordiCHI '08, pages 463–466, New York, NY, USA, 2008. ACM.

[GL96] H. Günther and O. Ludwig. *Schrift und Schriftlichkeit: ein interdisziplinäres Handbuch internationaler Forschung.* Number Bd. 2 in Handbooks of linguistics and communication science. New York, 1996.

[Han05] Jefferson Y Han. *Low-cost multi-touch sensing through frustrated total internal reflection.* ACM, New York, New York, USA, October 2005.

[HCW14] Sue Hessey, Szu Han Chen, and Catherine White. Beyond Fingers and Thumbs – A Graceful Touch UI. In *Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments*, pages 562–573. Springer International Publishing, Cham, June 2014.

[HG14] J Han and N E Gold. Lessons Learned in Exploring the Leap Motion™ Sensor for Gesture-based Instrument Design. *In: Caramiaux, B and Tahiroğlu, K and Fiebrink, R and Tanaka, A, (eds.) Proceedings of the Internation Conference on New Interfaces for Musical Expression. Goldsmiths University of London: London, UK. (2014)*, June 2014.

[JA14] Rabia Jafri and Syed Abid Ali. A GPS-Based Personalized Pedestrian Route Recording Smartphone Application for the Blind. In *HCI International 2014 - Posters' Extended Abstracts*, pages 232–237. Springer International Publishing, Cham, June 2014.

[Leaa] Leap, Inc. How does the Leap Motion Controller work. http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/. zuletzt besucht am: 31.03.2016.

[Leab] Leap Motion, Inc. Leap Motion. https://www.leapmotion.com. zuletzt besucht am: 19.1.2016.

[Leac] Leap Motion, Inc. Leap Motion App Store - Touchless for Mac. https://apps.leapmotion.com/apps/touchless-for-mac/osx. zuletzt besucht am: 19.1.2016.

[Lead] Leap Motion, Inc. Leap Motion Developer Gallery - User: metalcorehero. https://developer.leapmotion.com/gallery/touch-everything. zuletzt besucht am: 19.1.2016.

[Mic] Microsoft. Microsoft Surfaces. https://www.microsoft.com/surface/de-de. zuletzt besucht am: 31.03.2016.

[Mos12] Johannes Moser. Konzept, Gestaltung und prototypenhafte Implementierung eines Multimedia Authoring Systems für Blinde. Master's thesis, Fakultät für Informatik der Technischen Universität Wien, March 2012.

[MTF10] Christian Müller-Tomfelde and Morten Fjeld. Introduction: A Short History of Tabletop Research, Technologies, and Products. In *Tabletops - Horizontal Interactive Displays*, pages 1–24. Springer London, 2010.

[Nah15] Vandad Nahavandipoor. *iOS 9 Swift Programming Cookbook*. Solutions and Examples for iOS Apps. Ö'Reilly Media, Inc.", December 2015.

[Neu15] Matt Neuburg. *IOS 9 Programming Fundamentals with Swift*. Swift, Xcode, and Cocoa Basics. Ö'Reilly Media, Inc.", September 2015.

[Par] Parse, Inc. iOS Guide - Getting Started. https://parse.com/docs/ios/guide. zuletzt besucht am: 23.03.2016.

[PCZL13] A Paladugu, P S Chandakkar, P Zhang, and B Li. Supporting navigation of outdoor shopping complexes for visuallyimpaired users through multi-modal data fusion. *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–7, 2013.

[PMPRG11] Benjamin Poppinga, Charlotte Magnusson, Martin Pielot, and Kirsten Rassmus-Gröhn. *TouchOver map: audio-tactile exploration of interactive maps*. audio-tactile exploration of interactive maps. ACM, New York, New York, USA, August 2011.

[PSWP13] S Pölzer, D Schnelle-Walka, and D Pöll. Making brainstorming meetings accessible for blind users. *AAATE ...*, 2013.

[Saf08] D. Saffer. *Designing gestural interfaces*. O'Reilly Media, Inc., 2008.

[SBD08] J Schöning, P Brandl, and F Daiber. Multi-touch surfaces: A technical guide. *IEEE Tabletops and ...*, 2008.

[SdAdA13] E S Silva[1], JAO de Abreu[1], and JHP de Almeida. A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments. 2013.

[Sen] Sensel, Inc. Multi-Touch, Pressure Sensitive Technology. `http://www.sensel.com`. zuletzt besucht am: 30.3.2016.

[SHBS10] J Schöning, J Hook, T Bartindale, and D Schmidt. Building interactive multi-touch surfaces. *Tabletops-Horizontal ...*, 2010.

[Som13] Paul Sommersguter. Interaction Design for the Blind. Master's thesis, Fakultät für Informatik der Technischen Universität Wien, February 2013.

[Str10] Strukt GmbH. Struktable. `http://strukt.com/2010/struktable/`, 2010. zuletzt besucht am: 31.3.2016.

[SW14] Erica Sadun and Rich Wardwell. *The Core iOS Developer's Cookbook*. Addison-Wesley Professional, March 2014.

[SWAO+14] Dirk Schnelle-Walka, Ali Alavi, Patrick Ostie, Max Mühlhäuser, and Andreas Kunz. A Mind Map for Brainstorming Sessions with Blind and Sighted Persons. In *Computers Helping People with Special Needs*, pages 214–219. Springer International Publishing, Cham, July 2014.

[Tha] Thalmic Labs Inc. Gesture Control Armband. `https://www.myo.com`. zuletzt besucht am: 31.03.2016.

[Thi] Thingiverse. iPad Pro Case - User: fflood. `http://www.thingiverse.com/thing:1212566`. zuletzt besucht am: 27.02.2016.

[VZ14] Radu-Daniel Vatavu and Ionut-Alexandru Zaiti. Leap gestures for TV: insights from an elicitation study. In *TVX '14: Proceedings of the 2014 ACM international conference on Interactive experiences for TV and online video*, pages 131–138, New York, New York, USA, June 2014. ACM Request Permissions.

[WNL06] Bruce N Walker, Amanda Nance, and Jeffrey Lindsay. Spearcons: speech-based earcons improve navigation performance in auditory menus. 2006.