Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Sensorbasierte Kompensierung von Eingabelatenzen auf Touchscreens

Philipp Göttlich

Studiengang:	Informatik
Prüfer/in:	Jun.-Prof. Dr. Niels Henze
Betreuer/in:	Dipl.-Ing. (FH) Valentin Schwind, Huy Viet Le, M.Sc.
Beginn am:	27. Oktober 2016
Beendet am:	27. April 2017
CR-Nummer:	H.5.2.

Kurzfassung

Touchscreens sind eine weit verbreitete Schnittstelle zwischen Mensch und Computer, vor allem bei mobilen Geräten. Ein Nachteil gegenüber anderen Eingabemethoden stellt allerdings die Latenz zwischen einer Toucheingabe und der darauf zugeschnittenen, visuellen Reaktion dar, welche oft in deutlich spürbaren Bereichen liegt. Latenzen verschlechtern nachweislich die Performanz und Benutzererfahrung der Anwender. Aus diesem Grund haben sich viele vorangegangene Arbeiten bereits mit der Reduzierung solcher Latenzen beschäftigt, was auch Gegenstand dieser Arbeit ist. Ein vorheriger Ansatz verwendete eine Extrapolation von Fingerbewegungen, um mit Hilfe von künstlichen, trainierten, neuronalen Netzen Toucheingaben zur Latenzkompensation vorherzusagen. Aufbauend auf dem teilweisen Erfolg der erwähnten Arbeit, wurde diese Idee im vorliegenden Ansatz aufgegriffen und mit Hilfe von inertialen Messeinheiten (IMUs) eines eigens entwickelten, hochfrequenten Sensorprototyps verbessert. Die entstandenen, auf Touchtrajektorien und IMU-Daten trainierten, neuronalen Netze konnten Genauigkeitsverbesserungen bei Vorhersagen von bis zu 30% gegenüber IMU-losen Netzen aufweisen. Des Weiteren konnte mit diesen Netzen der Durchsatz von Nutzern um 15% bei Fingereingaben und um 17% bei Stifteingaben gesteigert werden, wenn die Anwender eine IMU an der Hand trugen.

Abstract

Touchscreens are a common interface between humans and computers, especially in mobile devices. However, a disadvantage compared to other input methods is the latency between a touch input and the visual response to this, which often lies in clearly perceptible areas. Latencies are proven to deteriorate the performance and user experience of users. For this reason, a wide range of previous work has already dealt with the reduction of such latencies, which is also the subject of this work. A previous approach used an extrapolation of finger movements to predict its future position for latency compensation using artificial, trained, neural networks. Based on their partial success, this idea was taken up in the present approach and improved with the aid of inertial measuring units (IMUs) of a custom, high-frequency sensor prototype. The resulting neural networks trained on touch trajectories and IMU data have been able to improve the accuracy of predictions of up to 30% against IMU-free networks. In addition, the networks enabled users to increase their throughput by 15% for finger inputs and 17% for pen inputs when users were wearing an IMU on their hand.

Inhaltsverzeichnis

1 Einleitung

Touchscreens sind seit der Erfindung des Smartphones eines der meistverbreiteten Ein- und Ausgabegeräte im täglichen Gebrauch. Sie sind allgegenwärtig und werden in vielen verschiedenen Bereichen für unterschiedliche Aufgaben verwendet. Obwohl sie ständig weiterentwickelt werden, haben Touchscreens meist eine höhere Eingabelatenz als andere Eingabegeräte, wie etwa Computermäuse. Die Eingabelatenz beschreibt dabei die Zeit zwischen einem Stimulus und einer sichtbaren Reaktion auf dem Gerät. MacKenzie und Ware beschrieben Latenz als die Zeit zwischen einer Eingabeaktion und einer Ausgabereaktion darauf [MW93].

Nicht selten liegt eine solche Latenz bei Touchscreens im Bereich von 100ms, was dazu führt, dass viele Eingaben sich nicht intuitiv anfühlen und die Performanz der Benutzer bei entsprechenden Aufgaben beeinträchtigt wird [Deb+16; Ng+12]. Bei schnellen, kontinuierlichen Touchbewegungen, über längere Distanzen, lässt sich die Latenz visuell betrachten. Zwischen dem Eingabestimulus, also der Fingerberührung, und der Reaktion auf dem Touchscreen bildet sich aufgrund der Latenz oft ein positioneller Abstand. Je schneller und ausschweifender die Touchbewegung, desto größer wird dieser Abstand zwischen Finger und visueller Reaktion. Bei Zieh- oder Zeichenbewegungen auf einem Touchgerät (Im Zusammenhang mit dieser Arbeit bezeichnet dieser Begriff Smartphones, Tablets oder andere Geräte mit Touchscreen) kann der Abstand mehrere Zentimeter betragen, was dazu führt, dass das manipulierte Objekt zum Beispiel dem Finger sichtlich hinterher zieht. Dieses Verhalten wurde in Abbildung 1.1 am Beispiel einer Linienzeichnung dargestellt. Nutzer interpretieren dies oft als Fehlverhalten oder Ungenauigkeit bei der Reaktion des Gerätes. Dieses Gefühl wird bei der Nutzung von Stiften zur Toucheingabe, sogenannten Styli (Singular: Stylus), noch verstärkt, da diese einen geringeren Teil des Bildschirms verdecken als ein Finger. Die visuellen Auswirkungen von Latenzen sind dadurch leichter wahrzunehmen.

Die Folgen sind Frustration der Nutzer und Fehler bei zeitkritischen Anwendungen, die so weit gehen können, dass Anwender auf alternative Touchgeräte oder sogar völlig andere Eingabegeräte umsteigen. Auch für die Hersteller von Touchscreens oder zugehörigen Geräten wird die Latenz somit zu einem großen Problem, um sich auf dem Markt zu behaupten. Der resultierende Konkurrenzkampf sorgt zwar für eine fortschreitende Verbesserung von Touchscreenlatenzen, jedoch befinden sich diese bei fast allen kommerziellen Geräten immer noch in einem deutlich spürbaren Bereich, was viele Aufgaben anbelangt, besonders bei der Benutzung von Styli zur Eingabe [Jot+13; Ng+12]. Einige vorangegangene Arbeiten haben gezeigt, dass sich Latenzen von über 25ms negativ auf die Performanz der Nutzer auswirken und sogar Latenzen von nur 10ms spürbar sind [Jot+13; Ng+12]. Bei aktuellen Touchgeräten beträgt die

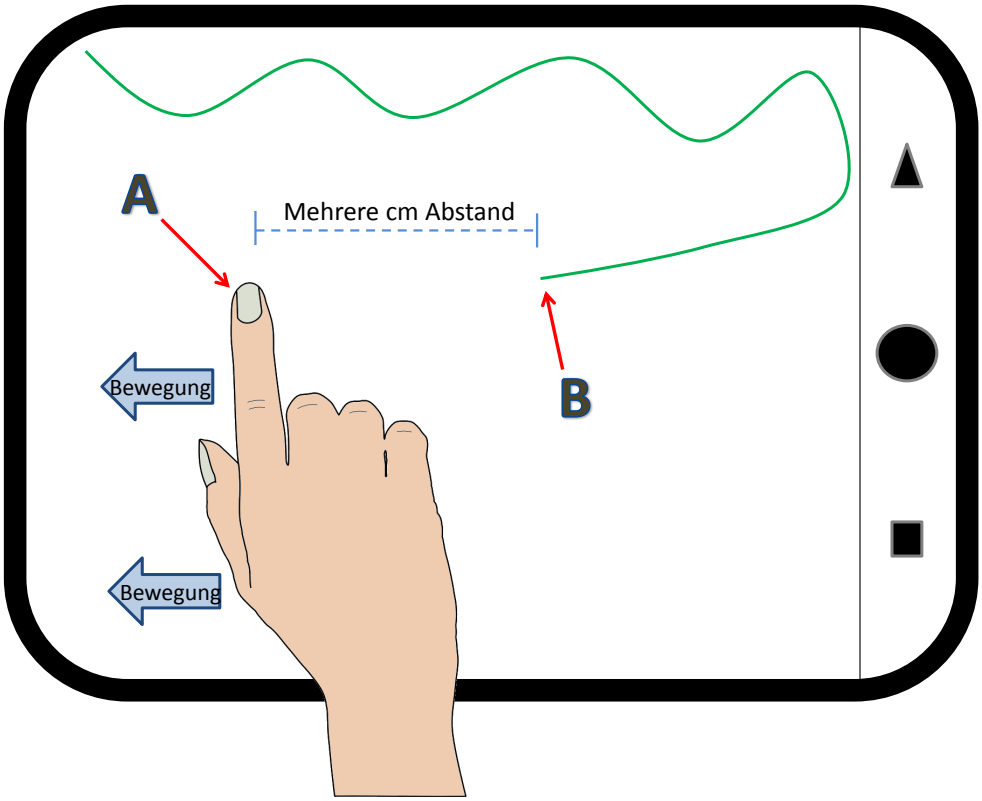


Abbildung 1.1: Die gezeichnete Linie zieht dem Finger auf einem Touchgerät sichtlich hinterher. Zwischen Fingerposition (A) und Touchreaktion/Zeichnung (B) bildet sich ein Abstand von mehreren Zentimetern.

Latenz meistens zwischen 50ms und 200ms [Ng+12]. Sie ist also für die meisten Benutzer spürbar und auch bei unbewusster Wahrnehmung fühlen sich die zugehörigen Geräte langsamer oder weniger intuitiv an. Eine Kompensation der Eingabelatenz von Touchscreens würde sich demnach sowohl auf die Performanz, als auch auf das intuitive Arbeiten beim Verwenden von Touchscreens auswirken, was ebenfalls Frustration bei Anwendern vorbeugt.

Das komplette Umgehen von Latenzen ist unmöglich, da die Verarbeitung zwischen Ein- und Ausgabe durch Hardware und Software zusätzliche Latenzen erzeugt, selbst wenn die Abtastrate der Eingabe oder die Ausgaberate des Bildschirms sehr hoch sind [MW93]. Das Kompensieren/Reduzieren von Latenzen wurde deshalb zu einem populären Forschungsgegenstand in der letzten Zeit. Viele Ansätze, die sich damit beschäftigen und geringe Latenzen aufweisen, sind schlichtweg zu teuer oder zu groß, um in kommerziellen Geräten Anwendung zu finden. Sie setzen oft auf teure externe Hardware mit hohem Stromverbrauch, wie etwa externe Kameras für Hochgeschwindigkeitsmessungen [Ng+12]. Solche Ansätze können eventuell bei zeitkritischen Aufgaben in der Industrie Anwendung finden, oder in Zukunft zum Einsatz kommen, wenn die nötige Hardware günstiger oder kleiner geworden ist. Andere Ansätze, wie die Arbeit von Henze et al., die sich darauf beschränken, Latenzen ausschließlich

durch Software zu reduzieren, versprechen aber ebenfalls bereits gute Ergebnisse, wobei der Reduzierungsgrad geringer ausfällt [HFS16]. Die Autoren des zuletzt genannten, rein softwarebasierten Ansatzes konnten bisher nur eine Reduzierung der Latenz um 66ms zeigen, was aber bereits die Performanz und Benutzererfahrung der Anwender verbesserte. Dafür wurde eine Vorhersagetechnik verwendet, die Nutzereingaben auf Basis vorangegangener Eingaben durch Verwendung neuronaler Netze vorhersagt. Der Nachteil dabei ist, dass durch diese Methode ungewollte visuelle Effekte auftreten, die bei Nutzern auf große Ablehnung stoßen. Diese visuellen Effekte werden als „Jitter“ bezeichnet, was sich mit „Schwankungen“ übersetzen lässt. Es handelt sich dabei um ein sichtliches Zittern der Eingabe welches als Störsignal wahrgenommen wird und deshalb unerwünscht ist.

Diese Arbeit hat das Ziel, eine Kompensierung von Latenzen auf Touchscreens durch die Kombination von Vorhersage durch Software, wie bei der Arbeit von Henze et al. [HFS16], mit aktueller Hardware zu kombinieren. Bildlich gesehen soll der hier vorgestellte Ansatz die in Abbildung 1.1 dargestellte Lücke schließen und dabei die Vorteile hardware- und softwarebasierter Ansätze kombinieren, ohne die Nachteile in Kauf nehmen zu müssen. Als Vorlage für die softwarebasierte Komponente dient die Arbeit von Henze et al. [HFS16]. Die dort genutzten neuronalen Netze sollen durch zusätzliche inertiale Messeinheiten (englisch: inertial measurement unit, Abkürzung: IMU) verbessert werden, welche verschiedene Sensoren kombinieren und normalerweise im Bereich der Navigation verwendet werden.

Die Intention bei der Verwendung von inertialen Messeinheiten ist, dass diese der Vorhersage mehr Genauigkeit ermöglichen, indem sie Informationen zur Bestimmung der Handbewegungen der Anwender in hoher Frequenz zur Verfügung stellen. Dazu gehören vor allem die Beschleunigung der Hand und die Ausrichtung der Hand im dreidimensionalen Raum über die Zeit. Theoretisch helfen die Messeinheiten folglich dabei, die visuellen Effekten von Vorhersagen rein softwarebasierter Ansätze zu reduzieren, indem mehr Genauigkeit über gemessene Handbewegungen vermittelt wird. Dadurch verbessert sich die Benutzererfahrung der Anwender gegenüber rein softwarebasierter Ansätze, wie der Arbeit von Henze et al. [HFS16], da die visuellen Effekte von den Nutzern dort als sehr störend wahrgenommen wurden. Fällt diese Verbesserung stark aus, so ließe sich die Kompensierung der Touchscreenlatenz auch über 66ms hinaus nutzen, was bei einigen Geräten sogar zur vollständigen Entfernung der Latenz führen würde. Es sollen sich durch das neue System sowohl die Benutzererfahrung der Anwender, als auch deren Performanz, aufgrund von weniger Latenz, verbessern.

Die inertialen Messeinheiten kommen bereits in fast allen kommerziellen Touchgeräten vor, wie etwa in Smartphones, Tablets oder Smartwatches und sind dabei sehr klein und preiswert. Die Nachteile anderer hardwarebasierter Ansätze fallen damit weg. Die zukunftsorientierte Idee, welche die Nutzung solcher inertialen Messeinheiten unterstützt, ist, dass bereits viele Menschen Geräte mit inertialen Messeinheiten im täglichen Gebrauch verwenden, wie etwa Smartphones und Smartwatches. Interessant für diese Arbeit sind vor allem Smartwatches. Die darin enthaltenen Sensoren können genutzt werden, um die Latenz der Touchscreens anderer gekoppelter Geräte, wie etwa Smartphones oder Tablets zu reduzieren, ohne dass zusätzliche Hardware benötigt wird. Folgt man diesem Ansatz weiter, so kommen auch Smartrings oder

Smartpens in den Sinn, welche auch inertiale Messeinheiten enthalten können und sich dadurch für diesen Ansatz eignen.

Das übergreifende Ziel ist demnach, dass Touchscreennutzer in Zukunft ihre bereits verfügbaren Geräte, wie Smartwatches, Smartrings oder Touchpens nutzen können, um die Latenz auf anderen Touchscreens effektiv und ohne entstehenden Aufwand oder zusätzliche Kosten reduzieren zu können. Durch die aktuelle Verbreitung von Smartwatches stellt dieser Ansatz eine erfolgversprechende Alternative zu bisherigen Latenzkompensierungsansätzen dar. Diese Eigenschaft kann auch in Zukunft weiter gewährleistet werden, wenn Smartrings vermehrt auf den Endverbrauchermarkt kommen [Mai17].

Zur Erreichung des gesetzten Ziels, der Erstellung eines kombinierten Systems zur Latenzkompensierung, welches Benutzererfahrung und Performanz verbessert, müssen mehrere Teilziele erreicht werden. In einem ersten Schritt muss ein Hardware-Testsystem entwickelt werden, welches externe inertiale Messeinheiten an ein Touchgerät anbindet und deren Sensordaten übermittelt. Die genutzte Sensorik soll dabei nicht die einer modernen Smartwatch übersteigen. Außerdem müssen die Messeinheiten auf der Hand der Nutzer dort platziert werden, wo normalerweise auch die zu imitierenden Smartgeräte (Dieser Begriff wird im Verlauf der Arbeit als Zusammenfassung der folgenden, mit IMUs ausgestatteten Geräte verwendet: Smartwatches, Smartrings und Smartpens) sitzen würden, also Handgelenk, Finger und Stylus, wie in Abbildung 1.2 dargestellt. Die dargestellten Bezeichnungen der IMU-Positionen dienen im Verlauf dieser Arbeit zur Identifikation der einzelnen IMUs. Die Fingerposition entspricht nicht exakt der Trageposition eines Smartrings, erleichtert allerdings die Anbringung der später beschriebenen IMUs erheblich und weicht positionell nur leicht von einer realen Smartring-Trageposition ab.

Auf dieses Weise kommt das Verhalten des Prototypen dem späteren Verhalten mit kommerziellen Geräten nahe. Die Daten, welche das Hardware-Testsystem liefert, sollen folgende Kriterien möglichst genau einhalten: Die Daten müssen synchron, hochfrequent und fehlerfrei sein. Dazu müssen passende Komponenten, wie IMUs und Mikrocontroller, gewählt und konfiguriert werden. Sind die externen Inertialsensoren den Kriterien entsprechend gewählt und angebunden, so beginnt der zweite Schritt. Eine Studie wird durchgeführt, bei welcher Nutzer, mit dem Hardware-Testsystem ausgestattet, einige Aufgaben auf einem Touchgerät durchführen müssen. Dabei werden Touch- und Sensordaten gesammelt und gespeichert.

Der dritte Schritt besteht in der Erstellung mehrerer neuronaler Netze mit den gesammelten Touch- und Sensordaten, ähnlich wie in der Arbeit von Henze et al. [HFS16]. Von diesen werden anschließend einige ausgewählt, welche im darauf folgenden Schritt getestet werden. Es folgt eine weitere Studie, mit den ausgewählten neuronalen Netzen als Modelle zur Vorhersage. Dabei wird die Performanz der Nutzer als Durchsatz mit einem Fitts' Law-Test [Fit54; Mac92] und ihr subjektives Empfinden als Benutzererfahrung gemessen. Am Ende folgt die Evaluation der Ergebnisse.

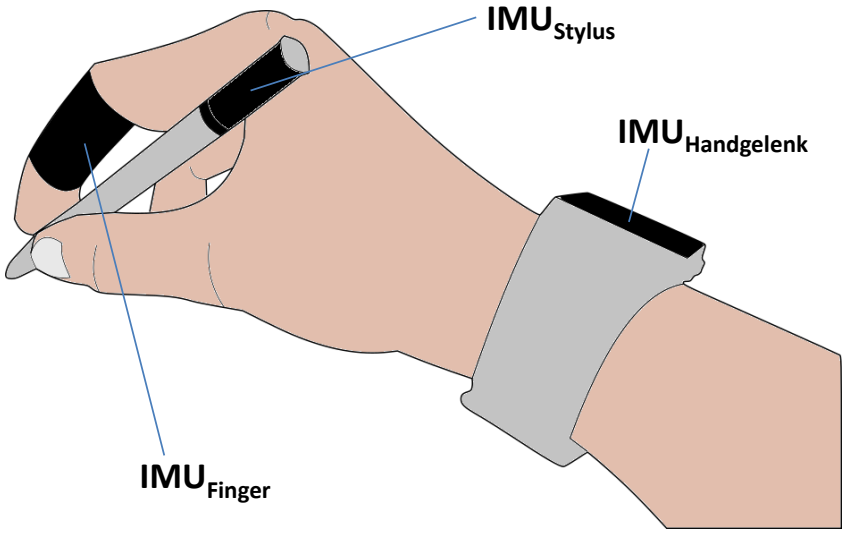


Abbildung 1.2: Diese Abbildung demonstriert die drei IMU-Positionen, die für diese Arbeit ausgewählt wurden. Sie entsprechen, bis auf die Fingerposition, welche leicht abweicht, den typischen Tragepositionen von Smartgeräten, welche mit IMUs ausgestattet sind, wie etwa Smartwatch, Smartpen und Smartring.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Hintergrund: Hier werden die Grundlagen dieser Arbeit beschrieben.

Kapitel 3 – Anbindung externer Inertialsensoren: Hier wird die Entwicklung des in der Arbeit verwendeten, externen Hardwareprototyps zur Sensoranbindung dargestellt.

Kapitel 4 – Studie zur Datenerhebung: Dieses Kapitel widmet sich der Erhebung von Trainingsdaten für die neuronalen Netze und der dafür durchgeführten Studie.

Kapitel 5 – Neuronale Netze: In diesem Abschnitt werden mit den gesammelten Daten des vorherigen Kapitels neuronale Netze erstellt, auf welche im Detail eingegangen wird.

Kapitel 6 – Evaluation der Modelle: Inhaltlich werden hier sowohl die Methoden der Evaluation, der im vorherigen Kapitel erstellten Modelle, mit Hilfe einer weiteren Studie, erläutert, als auch deren Ergebnisse vorgestellt und diskutiert.

Kapitel 7 – Zusammenfassung und Ausblick: Dieses abschließende Kapitel fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2 Hintergrund

Der Hintergrund dieser Arbeit setzt sich aus den verwandten Arbeiten zum Thema Latenzen und den Grundlagen über inertiale Messeinheiten zusammen, da diese beiden Gebiete in der vorliegenden Arbeit zusammengeführt werden. Zuerst werden die verwandten Arbeiten beschrieben, die den Rahmen für diese Arbeit aufspannen. Darin werden besonders Grundlagenarbeiten zum Thema Latenzen erwähnt, aber auch konkrete Ansätze, um Latenzen zu reduzieren. Im zweiten Teil werden dann die Inertialsensoren vorgestellt, welche im Laufe dieser Arbeit verwendet wurden und kurz deren Funktionsweise und Relevanz für diese Arbeit erläutert.

2.1 Verwandte Arbeiten

Der Hintergrund dieser Arbeit baut auf einigen vorangegangen Arbeiten mit verschiedenen Schwerpunkten auf. Die darin gewonnen Erkenntnisse werden aufgegriffen, kategorisiert und diskutiert, um den Rahmen für diese Arbeit zu bestimmen. Dazu werden zuerst bekanntere Arbeiten erwähnt, welche sich bereits mit Latenzen in der Mensch-Computer-Interaktion beschäftigten, um die Definition, Wahrnehmung und Folgen von Latenz darzustellen. Anschließend wird auf konkrete Ansätze eingegangen, welche zum Ziel hatten, Latenzen zu kompensieren und dadurch als Orientierung für die vorliegende Arbeit dienen.

2.1.1 Latenz und deren Messung

Latenz beschreibt die Zeit zwischen einer Eingabe und der entsprechenden Antwort darauf. Bei Touchscreens ist dies die Zeit zwischen einer Berührung des Bildschirms und dem daraus resultierenden visuellen Feedback auf dem Bildschirm. Diese Verzögerung ist ein fester Bestandteil eines Systems aus Software, Eingabe- und Ausgabegeräten und damit an deren Eigenschaften gebunden [MW93]. Wie MacKenzie und Ware beschrieben, ist Latenz in einem solchen System unvermeidbar und hängt vor allem von den Frequenzen ab, mit welchen das Eingabegerät Daten abtastet und das Ausgabegerät sich aktualisiert [MW93]. Da das Zusammenspiel der einzelnen Komponenten sehr komplex ist und deren Aktualisierungsfrequenzen schwanken können, lässt sich die Latenz nicht einfach berechnen. Viele Arbeiten beschäftigen sich deshalb mit der zuverlässigen Messung von Latenzen auf Touchscreens.

Zwei Methoden kommen von Bérard und Blanch [BB13]. Bei ihrer ersten vorgestellten Methode wird eine Kamera genutzt, um die Latenzen zu messen. Die zweite Methode verzichtet, auf Kosten von Genauigkeit, auf jede externe Hardware und liefert trotzdem nur wenig schlechtere Ergebnisse, was die Genauigkeit der Messungen angeht. Bei dieser Methode werden nachzufahrende Strecken auf dem Touchscreen und Geschwindigkeiten fest vordefiniert, sodass die reale Fingerposition zu jedem Zeitpunkt abgeschätzt werden kann. Ähnlich funktioniert auch ein von Cattan et al. vorgestelltes Verfahren, welches aber bei Tests genauere Ergebnisse liefert, indem Latenzen durch Vorhersagen angenähert werden [CRCB15]. Eine gemeinsame Ungenauigkeit der drei Methoden stellt die ausführende Person dar, die die Eingabe für die Touchscreens erzeugt. Deber et al. bauten dafür ein Werkzeug, welches automatisch eine kapazitive Touchberührung auslöst und über einen Photodetektor die Zeit bis zum visuellen Feedback misst [Deb+16]. Dabei haben sie für aktuelle Touchgeräte, wie das Google Nexus 9 oder das Apple IPhone 6, Latenzen im Bereich von durchschnittlich 50ms bis 100ms gemessen.

Diese Zahlen werden von den Messungen von Kämäräinen et al. gestützt, die in ihrer Arbeit den Einfluss der Eingabelatenz für verschiedene Eingabemethoden auf die Gesamtlatenz verschiedener Aufgaben erforscht haben [Käm+16]. Frühere Arbeiten haben Latenzen von 50ms bis 200ms bei aktuellen Geräten festgestellt [Ng+12]. Kaaresoja et al. kommen dagegen, bei der Untersuchung von Touchscreenlatenzen bei Smartphones, nur auf minimal 75ms und maximal knapp über 150ms [KB10]. Je nach Gerät und Software variieren diese Latenzen der Eingabeverarbeitung bei Touchscreens mehr oder weniger stark. Bisherige Arbeiten haben aber gezeigt, dass diese Latenzen sehr oft im Bereich von 50ms bis 200ms liegen [Deb+16; KB10; Ng+12].

2.1.2 Wahrnehmung und Folgen

Die Frage die dabei aufkommt ist, ob Nutzer Latenzen in diesen Bereichen überhaupt wahrnehmen. Auch damit beschäftigen sich Arbeiten, vor allem in Bezug auf die Benutzererfahrung und Performanz, mit unterschiedlichen Eingabegeräten und Latenzen [Mar11]. Bei typischen Aufgaben für Touchscreens, wie Schieben, Ziehen oder Scrollen, empfinden Nutzer bei einer höheren Latenz, dass das Gerät langsamer reagiert [ADG11]. Bei Schreib- und Zeichenaufgaben nehmen Anwender Latenzen viel schwächer wahr [Ann+14]. Für diese Art von Aufgaben muss die Latenz bereits im Bereich von 50ms oder höher liegen, um wahrgenommen zu werden. Bei Schiebe-, beziehungsweise Ziehaufgaben, bei welchen ein Objekt über den Bildschirm bewegt wird, können viele Anwender bereits bei geringen Latenzen von 11ms die Verzögerungen wahrnehmen [Deb+15]. Ng et al. fanden in ihrer Studie heraus, dass viele Teilnehmer bei Ziehaufgaben zuverlässig Latenzen von 5ms bis 10ms erkennen können und einige Teilnehmer sogar Latenzenunterschiede von nur knapp 2ms wahrnehmen konnten [Ng+12].

Die obigen Arbeiten haben bewiesen, dass Menschen Eingabelatenzen deutlich wahrnehmen können. Eine daraus resultierende Frage ist, ob sich diese negativ auf die Performanz der Anwender auswirken. Diese Frage war ebenfalls bereits Forschungsgegenstand einiger vorangegangener Arbeiten. Für herkömmliche Eingabegeräte, wie etwa Computermäuse, konnte

schon eine verringerte Performanz und Genauigkeit der Nutzer bei erhöhten Eingabelatenzen nachgewiesen werden [PG12; PS11]. MacKenzie und Ware konnten einen Zusammenhang zwischen Verzögerung und Geschwindigkeit beim Erledigen von Aufgaben nachweisen [MW93]. Sie erklären, dass Latenzen die menschliche Performanz verschlechtern und als ernstzunehmender Flaschenhals für die Benutzerfreundlichkeit angesehen werden müssen. Jota et al. zeigten später mit einem Fitts' Law-Test für Ziehaufgaben, wie stark sich verschiedene Latenzen auf die Performanz auswirken [Jot+13]. Nach ihren Erkenntnissen gibt es signifikante Verbesserungen der Performanz bei absteigenden Latenzen bis runter zu 25ms. Selbst bei Latenzwerten kleiner als 25ms konnten bei einigen Personen noch Verbesserungen vermerkt werden. Vielfach wurde nachgewiesen, dass geringere Latenzen bei der Eingabe Vorteile, vor allem bei der Performanz der Anwender, mit sich bringen [Deb+15; JT14; Jot+13; MW93]. Besonders groß ist auch der Einfluss von Eingabelatenzen auf Videospiele mit schnellen Bewegungen, wie etwa Ego-Shooter. Das fanden Ivkovic et al. bei ihrer Arbeit zum Einfluss von Eingabegerät- und Ausgabegerätlatenzen heraus [Ivk+15].

Auch im Bereich der virtuellen Realität (englisch: Virtual Reality, Abkürzung: VR) wirken sich geringere Latenzen positiv auf die Performanz der Nutzer aus [Ell+97; SC05]. In der erweiterten Realität (aus dem Englischen: Augmented Reality) wirkt sich Latenz sogar nicht nur auf die Performanz der Nutzer aus, sondern auch auf ihre Wahrnehmung und Motorik [AMR04; All+01; Nel+00]. Allison et al. zeigen beispielsweise, dass Latenz in diesem Gebiet die Stabilität der Nutzer beeinträchtigen kann [All+01]. Bei Latenzen über 50ms hatten Nutzer mehr Probleme, Objekten in Head-mounted Displays visuell zu folgen [Nel+00]. Meehan et al. führten Tests in einer virtuellen Umgebung bei verschiedenen Latenzen durch und stellten fest, dass Teilnehmer bei hohen Latenzen, etwa 90ms, eine verzögerte Wahrnehmung aufwiesen und die virtuelle Umgebung eher als unwirklich wahrnahmen [Mee+03]. Die Vorteile geringerer Latenzen wurden mehrfach erforscht und gerade im Bereich der verzögerten Reaktion auf Eingaben, wie etwa bei Touchscreens, versprechen geringere Latenzen bessere Performanzen der Anwender. Wie von einigen Arbeiten untersucht, ist die Latenz bei Eingaben auf Touchscreens noch vergleichsweise hoch, weshalb diese ein hohes Potential für Verbesserungen aufweisen. Die Reduzierung von Latenzen ist aber auch in anderen Bereichen ein erstrebenswertes Forschungsziel.

2.1.3 Kompensierung und Reduzierung

Itoh et al. untersuchten die Latenz von Trajektorien in einem System der erweiterten Realität [Ito+16]. In ihrem Ansatz geht es darum, die Bewegungen realer Objekte, zum Beispiel von Tennisbällen, in Echtzeit vorherzusagen und die vorhergesagte Bewegung für die Anwender zu visualisieren. Dafür nutzten sie Head-Mounted Displays. Die Latenz in diesem System, zwischen Erkennung der Trajektorien und dem Vorhersagen des weiteren Verlaufs, lag bei etwa 100ms, was in der erweiterten Realität große Auswirkungen haben kann [AMR04; All+01; Nel+00]. Für die Kompensierung der Latenzen nutzten sie eine Vorhersage. Diese verschiebt

die vorhergesagte Trajektorienbewegung um 100ms in die Zukunft, so dass die Trajektorie ab der aktuellen Position des Objekts fortführend gezeichnet wird.

Asselborn beschäftigte sich hingegen mit der Reduzierung der Latenz bei der Interaktion mit greifbaren Objekten (aus dem Englischen: „tangible objects“) auf einem kapazitiven Touchscreens [Ass16]. Diese Objekte sind an bestimmte virtuelle Objekte gebunden und lösen beim Interagieren zugehörige Ereignisse auf dem darunterliegenden Touchscreen aus. Für die Vorhersagen wurden ein lineares Vorhersagemodell und eines mit polynomialer Regression genutzt. Dadurch erzielte er im subjektiven Feedback seiner Studienteilnehmer bessere Ergebnisse in Bezug auf die Benutzererfahrung. Seine Vorhersagemethoden kommen dabei nur mit Software aus, ohne den Gebrauch von externer Hardware. Dadurch leiden sie aber stärker an Ungenauigkeiten oder Fehlern der Vorhersagen. Eine andere Art der Eingabelatenz ist die Latenz beim Tapping, dem kurzen Berühren des Bildschirms, um etwa einen Knopf zu drücken oder eine Applikation auszuwählen. Mit dieser Art von Latenz beschäftigten sich Xia et al. in ihrer Arbeit [Xia+14]. Mit einer Hochgeschwindigkeitskamera nahmen sie die Handbewegung der Anwender auf und konnten mit einer Latenz, kleiner als eine Millisekunde, bestimmten, wann der Nutzer wo den Bildschirm berührte. Auf diese Weise konnten sie die Latenz beim Tapping auf weniger als eine Millisekunde reduzieren, wodurch es unmöglich für Menschen wird, die verbleibende Latenz wahrzunehmen.

Zwei andere Ansätze, die eher in die Richtung der vorliegenden Arbeit gehen, sind die Arbeiten von Ng et al. [Ng+12] und Leigh et al. [Lei+14]. Beide Arbeiten beschäftigen sich mit der Latenz bei Ziehbewegungen auf Touchscreens und reduzieren diese auf eine Millisekunde oder weniger. In beiden Ansätzen kommen ein Hochgeschwindigkeitsprojektor, welcher die Visualisierung übernimmt und ein hochfrequentes Eingabegerät zum Einsatz. Leigh et al. benutzten eine Unterlage mit einem simultanen, orthogonalen Multiplexverfahren und Ng et al. verwendeten einen hochfrequenten, resistiven Touchsensor [Lei+14; Ng+12]. Mit beiden Ansätzen lässt sich eine Latenz bei Ziehaufgaben von weniger als einer Millisekunde erreichen. Der Nachteil bei den letzten drei Methoden ist der hohe Bedarf an unhandlicher und teurer externer Hardware, weshalb sich diese Verfahren nicht für Privatnutzer mit kommerziellen Touchgeräten eignen.

Andere Verfahren, zum Beispiel die bereits erwähnte Arbeit von Asselborn [Ass16], die Latenzen durch Vorhersagen ohne Hilfe externer Hardware reduzieren, scheinen für den hier gewünschten Ansatz erfolgversprechender. Lank et al. sagten in ihrer Arbeit Gesteneingaben vorher, um die Performanz der Anwender zu verbessern [LCR07]. Für die Vorhersage beschränkten sie mögliche Bewegungen auf die Gesetze der menschlichen Kinematik. Pasqual und Wobbrock verwendeten eine Zuordnung kinematischer Muster für die Vorhersage von Endpunkten bei Mausbewegungen [PW14]. Ziber et al. beschäftigten sich mit demselben Problem, nutzten aber als Vorhersagemethode statistisches maschinelles Lernen [ZDB12].

Ein weiterer rein softwarebasierter Versuch, Latenzen zu verringern, kommt von Cattan et al. [Cat+15]. Sie nutzten auf Nutzer abgestimmte, kurzfristige lineare Vorhersagen, um Latenzen im Bereich von 25ms bis 75ms entgegenzuwirken. Wie bei dieser Arbeit, liegt der Fokus nicht auf der Vorhersage von Endpunkten, sondern ganzer Bewegungen (Trajektorien), weshalb Fitts' Law-Ziehaufgaben [SM04] verwendet wurden, um die Performanz der Nutzer

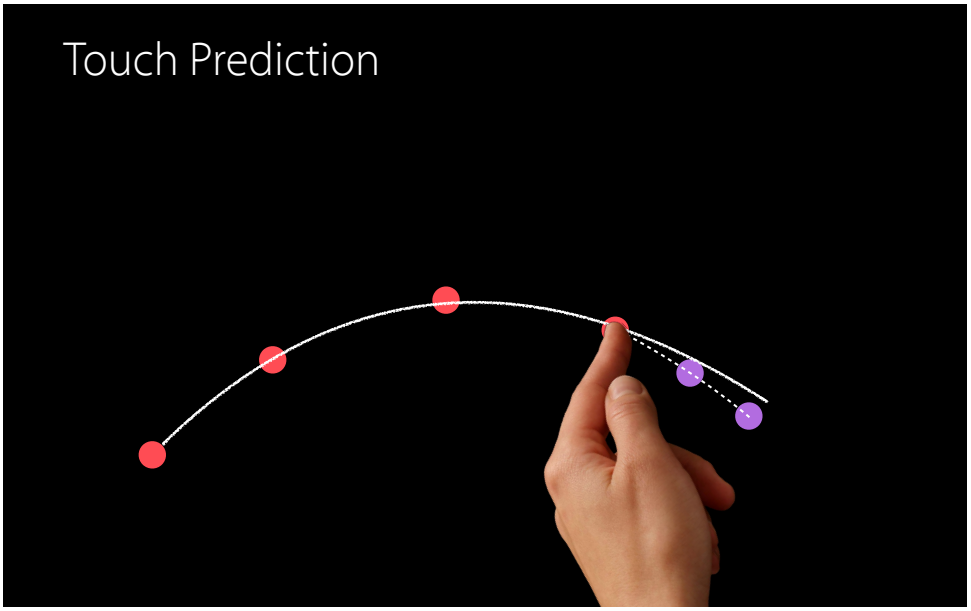


Abbildung 2.1: Das Bild zeigt die visuelle Darstellung der Trajektorienvorhersage auf Touchscreens von Apple für iOS, vorgestellt von Tsoi und Xiao 2015 [TX15] (Quelle ist ebenfalls Bildquelle). Die gestrichelte Linie demonstriert die vorhergesagte Trajektorie, welche leicht von der realen Bewegung abweicht.

mit kompensierter Latenz zu messen. Dabei fanden sie heraus, dass die Performanz der Nutzer sich bis zu einer Vorhersage/Kompensierung von 42ms verbesserte, darüber hinaus aber nicht mehr. Dies erklärten sie sich durch den höheren Vorhersagefehler und damit verbundenem „Zittern“ (englisch: Jitter; im weiteren Text wird der Begriff Jitter verwendet), unerwünschten visuellen Signalschwankungen, welche als Zittern wahrgenommen werden.

Diese Ungenauigkeiten und Zittereffekte sind Teil jeder Vorhersage und ihre Ausprägungen sind abhängig von der Weite und Art der Vorhersage, wie Vaidyanathan beschreibt [Vai07]. So wie Latenz die Performanz beeinflusst, hat Jitter einen Einfluss auf die Fehlerrate der Nutzer [PS09]. Es muss demnach ein Ausgleich gefunden werden, der sowohl die Latenz möglichst gut verringert, als auch den Jitter gering hält, wie Pavlovych und Stuerzlinger bei ihrer Arbeit herausstellten [PS09]. Trotz Jitter ist ein rein softwarebasierter Ansatz der Latenzkompensierung für den vorliegenden Anwendungsfall zu bevorzugen, da ein solcher auf jedem aktuellen Touchgerät implementiert werden kann.

Einige Touchgeräte der Firma Apple verfügen bereits, seit der Veröffentlichung des Betriebssystems iOS9, über eine Methode zur softwarebasierten Vorhersage von Trajektorienbewegungen [TX15]. Die Methode, visualisiert in Abbildung 2.1, wurde von Tsoi und Xiao 2015 ohne Details über die Realisierung vorgestellt.

Als Vorlage für die vorliegende Arbeit gilt der Ansatz von Henze et al. [HFS16]. Darin wird ein rein softwarebasierter Ansatz zur Kompensierung von Latenzen auf Touchscreens vorgestellt,

welcher die Position des Eingabegeräts (Finger, Stylus) über dem Touchscreen extrapoliert, um seine Position auf dem Touchscreen vorherzusagen. Anders als bei vielen der genannten Ansätze [LCR07; PW14; ZDB12] werden nicht nur Endpunkte vorhergesagt, sondern der Verlauf ganzer Trajektorien von Ziehbewegungen auf dem Bildschirm. Dazu wurden neuronale Netze mit den einzelnen Touchpunkten von Trajektorien trainiert, sodass zu jedem Zeitpunkt einer Touchbewegung der Verlauf mit Hilfe der letzten Berührungspunkte vorhergesagt werden kann. Die Art der Vorhersage ähnelt der von Cattan et al. [Cat+15], jedoch konnte durch Nutzung neuronaler Netze der Vorhersagefehler im Vergleich weiter verringert werden [HFS16]. Durch ein Fitts' Law-Ziehexperiment, mit den trainierten, neuronalen Netzen zur Vorhersage, konnten Henze et al. außerdem eine signifikant erhöhte Performanz der Nutzer untersuchen. Genutzt wurden dabei zwei Vorhersagestufen, von welchen die höhere Latenzen von 66ms und die niedrigere Latenzen von 33ms kompensierte.

Aufbauend auf der beschriebenen Arbeit, wird im Verlauf der vorliegenden Ausarbeitung versucht, den Vorhersagefehler bei der Vorhersage mit neuronalen Netzen weiter zu reduzieren. Dadurch sollen die Benutzererfahrung und Performanz noch weiter verbessert werden und eventuell sogar höhere Vorhersagestufen über 66ms ermöglicht werden.

2.1.4 Verwendung externer Sensorik

Biswas et al. veröffentlichten verschiedene Vorhersagemethoden für Punktvorhersagen, unter anderem mit der Verwendung neuronaler Netze, welche Geschwindigkeit, Beschleunigung, Ausrichtung und Distanz miteinbezogen [HP13]. Messwerten wie Beschleunigung und Ausrichtung könnten dabei auch für die Verbesserung der neuronalen Netze von Henze et al. genutzt werden und zur Verringerung von Vorhersagefehlern führen. Diese Daten lassen sich mit inertialen Messeinheiten messen, wie sie zum Beispiel in Smartwatches oder Smartphones vorkommen. Mit ihren Messdaten lassen sich einfach und präzise Bewegungen von Objekten verfolgen, weshalb sie ein großes Anwendungsgebiet besitzen [WF02]. Durch das Platzieren von inertialen Messeinheiten auf der Hand der Anwender sollen die nötigen Daten zur Verfolgung der Handbewegungen (englisch: Tracking) verfügbar gemacht werden, damit diese als Eingabe im neuronalen Netz dessen Vorhersagefehler reduzieren.

Das Nutzen von inertialen Messeinheiten zur Verbesserung existierender Systeme ist ein älteres Forschungsgebiet. Günthner beschreibt in seinem Buch Methoden, um kognitive Assistenzsysteme, wie sie etwa in Autos vorkommen, mit der Hilfe von inertialen Messdaten zu verbessern [Gün08]. Näher an der Thematik dieser Arbeit sind aber LaValle et al. mit ihrer Arbeit im Bereich der virtuellen Realität [LaV+14]. Sie verbesserten das Head-Tracking System der Oculus Rift VR-Brille[1] durch Bewegungsvorhersagen mit Hilfe von inertialen Messeinheiten. Durch ihr verbessertes System konnten sie eine bessere Benutzererfahrung untersuchen.

[1] Herstellerlink: https://www.oculus.com/rift/

2.1.5 Zusammenfassung

Zusammengefasst wurde viel über die Wahrnehmung und den Einfluss von Latenzen auf die Performanz geforscht. Da, besonders auf die Performanz der Nutzer bei Schiebe- und Ziehaufgaben, negative Einflüsse nachgewiesen wurden, ist es erstrebenswert, Latenzen zu kompensieren. Viele unterschiedliche Ansätze haben durch Latenzkompensierungen höhere Performanzen der Nutzer nachweisen können. Ein Ansatz von Henze et al. kompensiert Latenzen von bis zu 66ms durch Verwendung einer Vorhersage mit neuronalen Netzen. Da bei Vorhersagen gewisse Ungenauigkeiten auftreten, leiden die Benutzererfahrung und Fehlerraten der Anwender darunter. Mit Hilfe von inertialen Messeinheiten soll nun das Verfahren von Henze et al., im Hinblick auf eine Reduzierung der Vorhersagefehler, verbessert werden.

2.2 Sensoren

Sensoren werden genutzt, um verschiedene physikalische Größen in analoge Signale zu wandeln. Ein Beschleunigungssensor wandelt beispielsweise kontinuierlich Beschleunigung in einen analogen Zahlenwert um. Diese Werte stehen dann für eine Verarbeitung zur Verfügung. Zur Messung der realen physikalischen Größen liegt jedem Sensor ein Messprinzip zu Grunde, nach welchem er aufgebaut ist und welches seine Funktion definiert. Dieses Prinzip wurde in Abbildung 2.2 dargestellt. Sensoren verbinden also die reale Welt mit der digitalen und ermöglichen so eine Kommunikation über den Austausch von Messwerten.

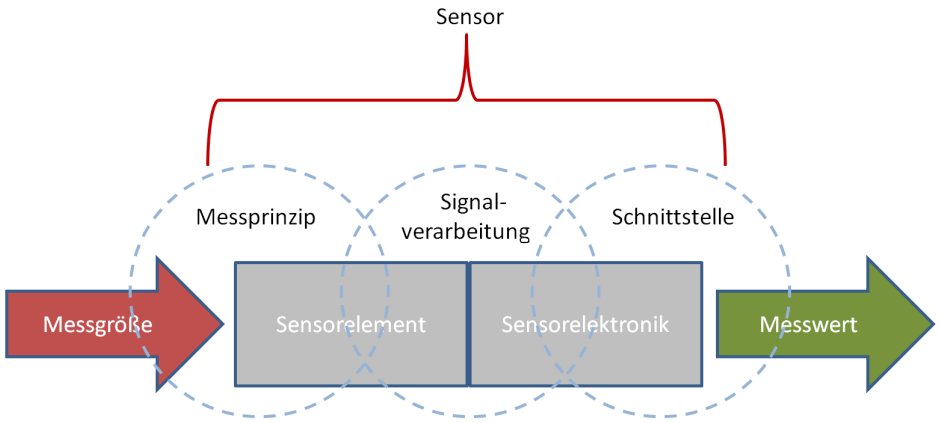


Abbildung 2.2: Ein im Sensorelement realisiertes physikalisches Messprinzip wird genutzt, um eine Messgröße in ein Signal umzuwandeln. Dieses Signal wird über die Elektronik des Sensors in ein digitales Signal gewandelt und weitergegeben, z.B. eine einfallende Lichtintensität als analoger Spannungswert für einen Lichtsensor [Ga14].

In den letzten Jahren haben sich Sensoren im Allgemeinen stark verändert. Neue Fertigungsprozesse ermöglichen die Produktion immer kleinerer und schnellerer Sensoren, welche durch eine bessere Abschirmung gegen Umwelteinflüsse noch genauere und zuverlässigere Daten liefern [Rei89; Sch13]. Der Trend der Miniaturisierung von Sensoren reduziert aber nicht nur Taktzeiten und Platzbedarf, sondern auch den Energiebedarf [Bie+99; Ga14]. Neue, effiziente, kompakte und leichte Sensoren lassen sich so in einer Vielzahl von neuen Anwendungsgebieten verwenden und ermöglichen Forschung und Automatisierung in vorher unwegsamen Bereichen [Ga14]. Die Anzahl genutzter physikalischer Effekte in Sensoren ist für Nutzer überwältigend [Ga14; Sch13], weshalb hier lediglich die relevantesten Effekte in ihrer einfachsten Form nach aktuellem Forschungsstand dieser Arbeit beleuchtet werden, welche in den verwendeten Sensoren genutzt werden.

In dieser Arbeit werden ausschließlich Inertialsensoren verwendet. Diese Sensoren sind dazu gedacht, die Lage, Position und Bewegung von Objekten zu erkennen, da sie häufig in Flugzeugen oder Schiffen zur Orientierung und Navigation verwendet werden. Häufig werden dazu gyroskopische Sensoren und Beschleunigungssensoren genutzt, zusätzlich können Magnetometer hinzugezogen werden [Flü10]. Die Kombination mehrerer Inertialsensoren, vor allem die Kombination eines gyroskopischen Sensors mit einem Beschleunigungssensor, nennt man inertiale Messeinheit, oder abgekürzt IMU (aus dem Englischen: „inertial measurement unit“) [Flü10]. Eine IMU besitzt Freiheitsgrade (englisch: degrees of freedom, Abkürzung: DoF), welche abhängig von den verbauten Sensoren sind. Ein Freiheitsgrad bezeichnet dabei eine Achse eines Sensors. Die meisten Sensoren von IMUs besitzen drei Achsen, um die Beschleunigung, Ausrichtung und Lage im dreidimensionalen Raum angeben zu können. Eine IMU mit einem dreidimensionalen gyroskopischen Sensor und einem dreidimensionalen Beschleunigungssensor hat demnach sechs Freiheitsgrade. Ist zudem ein dreidimensionales Magnetometer verbaut, besitzt die IMU folglich neun Freiheitsgrade. Die Freiheitsgrade werden bei der Bezeichnung einer IMU mit dem Kürzel „DoF“ angegeben, beispielsweise „9DoF-IMU“ für eine IMU mit drei dreidimensionalen Sensoren, also neun Freiheitsgraden. Im Zusammenhang mit der vorliegenden Ausarbeitung bezeichnet der Ausdruck „IMU“ stets eine Kombination aus den drei genannten Sensoren (Accelerometer, Gyroskop, Magnetometer) mit jeweils drei Achsen, welche im Folgenden genauer beschrieben werden.

2.2.1 Beschleunigungssensor

Beschleunigungssensoren, auch Accelerometer genannt, messen die Beschleunigung entlang ihrer Achsen. Werden sie fest mit einem Objekt verbunden, so ermöglichen sie die Messung der Beschleunigung des Objekts. Betrachtet werden hier nur mikroelektromechanische Accelerometer (MEMS Accelerometer). Solche kommen typischerweise auch in mobilen Geräten (Smartgeräten oder Touchgeräten) vor, zum Beispiel in Smartwatches oder Smartphones. Da die Funktionalität von Smartgeräten später nachgestellt werden soll, ist nur die Betrachtung von mikroelektromechanischen Accelerometern relevant.

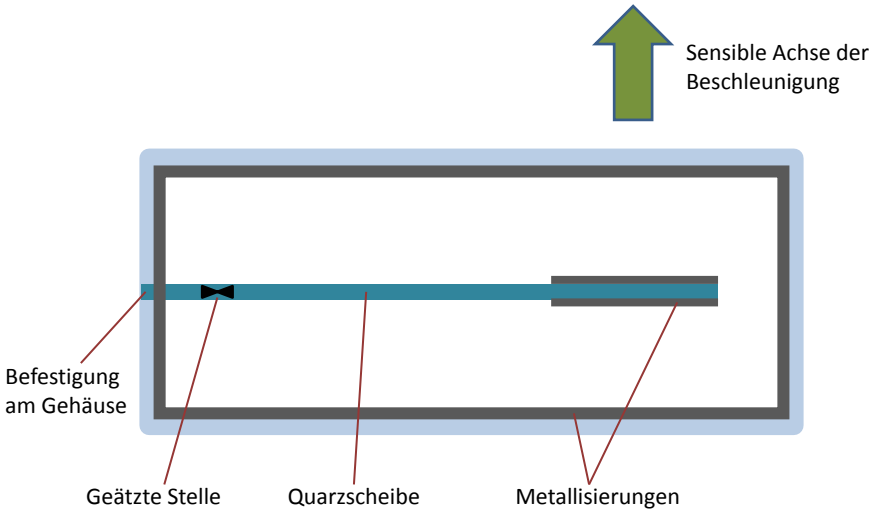


Abbildung 2.3: Zu sehen ist das Prinzip der Messung von Beschleunigungen entlang einer Achse [Flü10]. Dabei wird die Änderung einer Prüfmasse durch eine Beschleunigung entlang einer Achse kapazitiv gemessen.

Um die Beschleunigung zu messen, ist der Aufbau des Sensors ausschlaggebend. Für die Messung der Beschleunigung entlang einer Achse wird eine flexible Prüfmasse im Inneren des Sensors verbaut. Diese Prüfmasse ist oft eine Quarzscheibe, welche am Rand metallisiert wurde. An einer Stelle ist sie mit dem darum liegenden Gehäuse verbunden, welches ebenfalls im Inneren metallisiert wurde. Damit die Prüfmasse frei schwingen kann, wird sie knapp über der Stelle, an welcher sie mit dem Gehäuse verbunden wurde, geätzt, sodass sie das Verhalten einer Federlagerung aufweist. Dadurch wird die Prüfmasse flexibel und bewegt sich im Inneren des Gehäuses. Wird die Quarzscheibe nun entlang ihrer Normalen beschleunigt, so knickt die Scheibe an der geätzten Stelle ein und die Scheibe bewegt sich in Richtung einer der metallisierten Innenwände des Gehäuses. Dadurch findet zwischen den metallischen Komponenten eine Kapazitätsänderung statt, welche sich messen lässt [Du14; Flü10]. Es handelt sich bei Accelerometern also um kapazitive Sensoren.

Das Prinzip wurde in Abbildung 2.3 dargestellt. Die Lageänderung kann auf die Beschleunigung entlang der Achse zurückgerechnet werden [Flü10]. Um die Messwerte der Beschleunigung in alle drei Richtungen eines dreidimensionalen Raums zu erhalten, müssen drei Beschleunigungssensoren so orthogonal zueinander ausgerichtet werden, dass sie die drei Achsen für einen dreidimensionalen Raum aufspannen. In Abbildung 2.4 wurde dies schematisch dargestellt. Die Messwerte werden meist in m/s^2 angegeben. Die Sensitivität kann bei den meisten

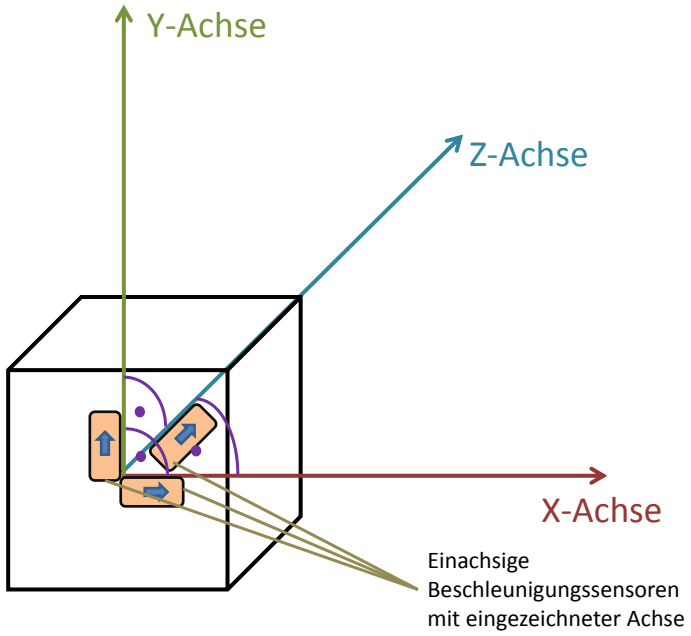


Abbildung 2.4: Drei einachsige Beschleunigungssensoren werden orthogonal zueinander angeordnet, so dass diese einen dreidimensionalen Raum aufspannen. Dieses Zusammenspiel ergibt einen dreiachsigen Beschleunigungssensor.

Accelerometern stufenweise in G-Kräften/Erdbeschleunigung (1G $\approx$ 9,81m/s^2) angegeben werden, wodurch die Werte automatisch auf dem entsprechenden Wertebereich vorskaliert werden. Aufgrund der zunehmenden Miniaturisierung, durch die Verwendung von Silizium und Dünnfilmelektroden als metallisierte Komponenten, ist die Größe der Sensoren aktuell kein kritischer Faktor. Die Kombination von drei einachsigen Beschleunigungsmetern stellt demnach kein Platzproblem dar [Ga14].

Der Nachteil bei der Verwendung von Accelerometern besteht in dem Versatz (englisch: Offset) der Messwerte, welcher sich über die Zeit der Messungen immer weiter vergrößert [Flü10]. Bei der Lagebestimmung von Objekten kann dieser Versatz der Beschleunigung in Kombination mit gyroskopischen Fehlern zu einer Lageabweichung von mehreren Kilometern pro Stunde führen [Flü10]. Der Versatz lässt sich aber über die Zuhilfenahme weiterer Messwerte schrittweise korrigieren, wie etwa über Magnetometerwerte, durch welche der Fehler über die Zeit stark verringert werden kann. Trotzdem ist die Abweichung zu groß, um exakte Positionen des Fingers auf einem Touchgerät zu bestimmen.

Das ist aber nicht relevant für das gegebene Vorhaben. Alleine über die Differenzen aufeinanderfolgender Beschleunigungsmessungen der Beschleunigung einer Hand, eines Stylus oder eines Fingers kann ein neuronales Netz lernen, in welche Richtung auf dem zweidimensionalen Bildschirm beschleunigt wird. Anschließend kann die Vorhersage auf diesen Bereich einge-

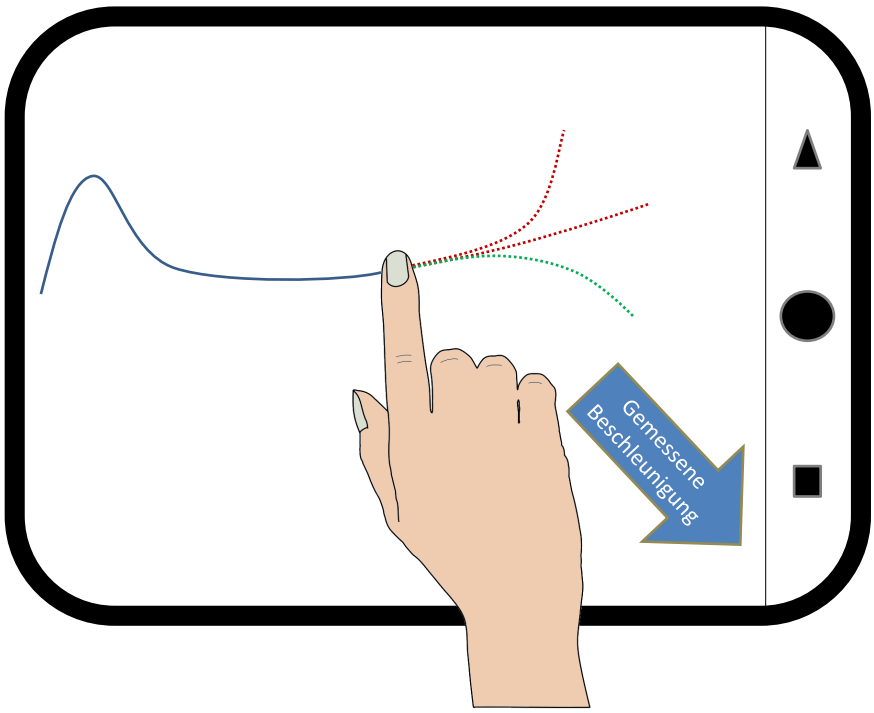


Abbildung 2.5: Ausgehend von einem aktuell gemessenen Touchpunkt gibt es drei mögliche Vorhersagen der Bewegung, die ein neuronales Netz aufgrund der vorherigen Trajektorie gelernt hat (gepunktete Linien) und deren Wahrscheinlichkeiten ungefähr gleich sind. Durch die aktuell gemessene Beschleunigung der Hand kommt nur die untere Vorhersage in Frage, welche dann ausgewählt wird (grüne Linie).

grenzt werden, was zu genaueren Vorhersagen führt. In Abbildung 2.5 wurde das gewünschte Verhalten visuell dargestellt. Begünstigt wird die Beschleunigung auch dadurch, dass sie fast ohne Latenz an das Gerät geliefert werden kann, also das Touchgerät mit aktuellen Daten versorgen kann. Das liegt daran, dass seit einigen Jahren die Frequenzen von mikroelektromechanischen Accelerometern, mit welchen die Daten aktualisert werden, bereits über 1kHz liegen [Alb+09; Du14]. Mit dem richtigen Aufbau sind Accelerometer mit solchen Frequenzen in der Lage, jede Millisekunde einen aktuellen Wert für die Beschleunigung zu liefern. Das ist vorteilhaft, um Latenzen auszugleichen, weil das Touchgerät selbst nur über veraltete Daten, bedingt durch die Latenz, verfügt. Das alles lässt vermuten, dass die Accelerometerwerte einen großen Einfluss auf die Genauigkeit der Vorhersagen haben. Demnach muss die für diesen Ansatz gewählte IMU über ein dreiachsiges Accelerometer verfügen, damit dies untersucht werden kann.

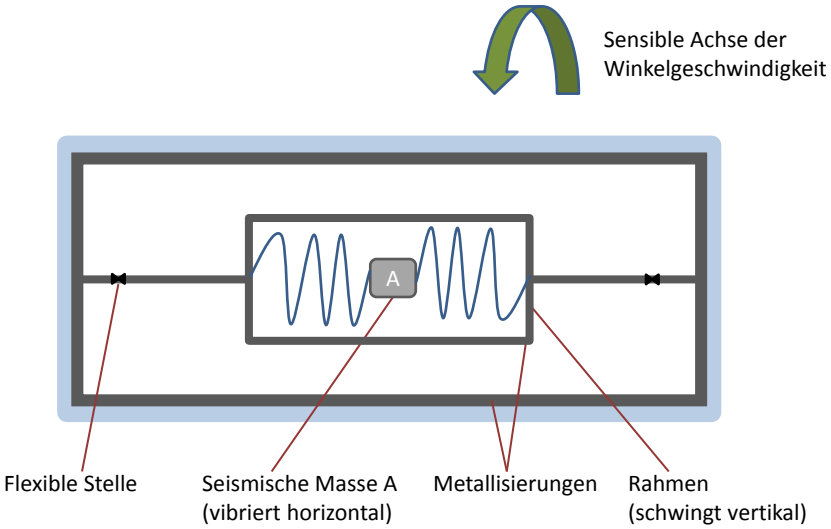


Abbildung 2.6: Eine seismische Masse A im Inneren des Sensors vibriert. Bei Drehbewegungen schwingt der flexible Rahmen darum senkrecht zur Bewegung von A, aufgrund der Corioliskraft. Die entstehenden Kapazitätsänderungen zwischen Rahmen und Gehäuse werden gemessen und sind proportional zur Winkelgeschwindigkeit entlang der Achse.

2.2.2 Gyroskop

Drehratensensoren, auch gyroskopische Sensoren oder Gyroskope, sind dazu gedacht, die Drehratenänderungen, auch Winkelgeschwindigkeiten, beziehungsweise damit verbunden die Rotationen entlang ihrer Achsen zu bestimmen. Durch die Platzierung auf einem Objekt ermöglichen sie die Orientierung des Objekts anzugeben. Auf diese Weise lässt sich die exakte Ausrichtung des Objekts im Raum berechnen. Sie werden oft im Bereich der Flugzeugtechnik verwendet [Flü10]. Die aktuell genausten Ausführungen sind der Laserkreisel (englisch: Ring Laser Gyro, Abkürzung: RLG) und der Faseroptische Kreisel (englisch: Fiber Optical Gyro, Abkürzung: FOG), welche in der Luftfahrt verwendet werden. Sie verwenden beide Lichtmessungsmethoden zur Bestimmung der Drehraten, was zwar sehr genau ist, aber die Sensorgröße derart erhöht, dass das Gyroskop zu groß wäre, um in ein Smartphone zu passen [Flü10].

Stattdessen werden in IMUs von Smartgeräten Kreisel sogenannter mikroelektromechanischer Systeme (Abkürzung: MEMS) verwendet [Ga14]. Diese sind um einiges kostengünstiger und kleiner. In dieser Arbeit werden nur solche Gyroskope verwendet. Die Funktionsweise ist dabei ähnlich, wie die der Beschleunigungssensoren. Eine flexible Prüfmasse wird durch Drehbewe-

gungen verschoben, wodurch Kapazitätsänderungen stattfinden, welche gemessen werden und auf Winkelgeschwindigkeiten umgerechnet werden können [Flü10]. Diese Gyroskope sind also gleichfalls kapazitive Sensoren [Du14]. Gleich sind auch die zunehmende Verwendung von Silizium und die daraus folgende Miniaturisierung der Strukturen [Ga14].

Anders ist allerdings das grundlegende physikalische Messprinzip. Die Corioliskraft wird genutzt, indem eine vibrierende Struktur Drehbewegungen ausgesetzt wird. Dadurch entstehen Schwingungen entlang der Normalen der Vibrationen, dessen Amplituden proportional zur Winkelgeschwindigkeit sind [Flü10]. Diese Amplitudenwerte werden, über Kapazitätsunterschiede gemessen, welche zwischen der vibrierenden Struktur und dem Gehäuse existieren. Das Prinzip ist in Abbildung 2.6 dargestellt. Die vibrierende Struktur, demnach gleichzusetzen mit der Prüfmasse, und das Gehäuse sind, wie beim Accelerometer, metallisiert. Die Werte der Winkelgeschwindigkeiten werden meist in rad/s angegeben und die Sensitivität der Sensoren lässt sich stufenweise in $Grad/s$ einstellen.

Nachteil ist auch hier der sich über die Zeit weiter verschlechternde Fehler der Messwerte. Dabei handelt es sich um die Abweichung (englisch: Drift) der Messwerte über die Zeit. Ebenfalls ist das Rauschen der Werte höher als bei optischen Kreiseln, weshalb diese sich nicht für hochpräzise Anwendungen eignen. Durch die Kombination mit anderen Sensoren, wie Magnetometern, lässt sich der Fehler reduzieren, wodurch zuverlässig und günstig die Trägheit von Objekten bestimmt werden kann [Flü10]. Die Winkelgeschwindigkeiten entlang aller drei Achsen eines dreidimensionalen Raumes lassen sich, wie beim Accelerometer, durch die orthogonale Anordnung dreier einachsiger Gyroskope erhalten, wie bereits in Abbildung 2.4 mit Accelerometern dargestellt.

Die Messwerte von Gyroskopen können später im neuronalen Netz für zusätzliche Genauigkeit sorgen, falls dieses die Werte der Accelerometer nicht passend mit dem Touchscreen verbindet. Durch die gyroskopischen Daten können die Accelerometerachsen auf die zweidimensionale Ebene des Touchscreens abgebildet werden, was dem neuronalen Netz diesen Lernschritt erspart. Dazu müssen zusätzliche gyroskopische Daten von dem Touchgerät selbst hinzugezogen werden, damit die Abbildung funktioniert. Wie beim Accelerometer liegen die Aktualisierungsfrequenzen der mikroelektromechanischen Gyroskope schon seit einiger Zeit über einer Frequenz von 1kHz, weshalb diese wahrnehmbar in Echtzeit zur Verfügung stehen [AAA07; Du14]. Wie stark sich allerdings die Abweichungen der Gyroskopwerte bei dieser Arbeit auswirken, muss noch untersucht werden. Nutzt das neuronale Netz lediglich Differenzen aufeinanderfolgender Werte, so sind keine Abweichungskorrekturen nötig. Dreiachsige Gyroskope sind also eine weitere Komponenten, die die hier verwendeten IMUs besitzen müssen.

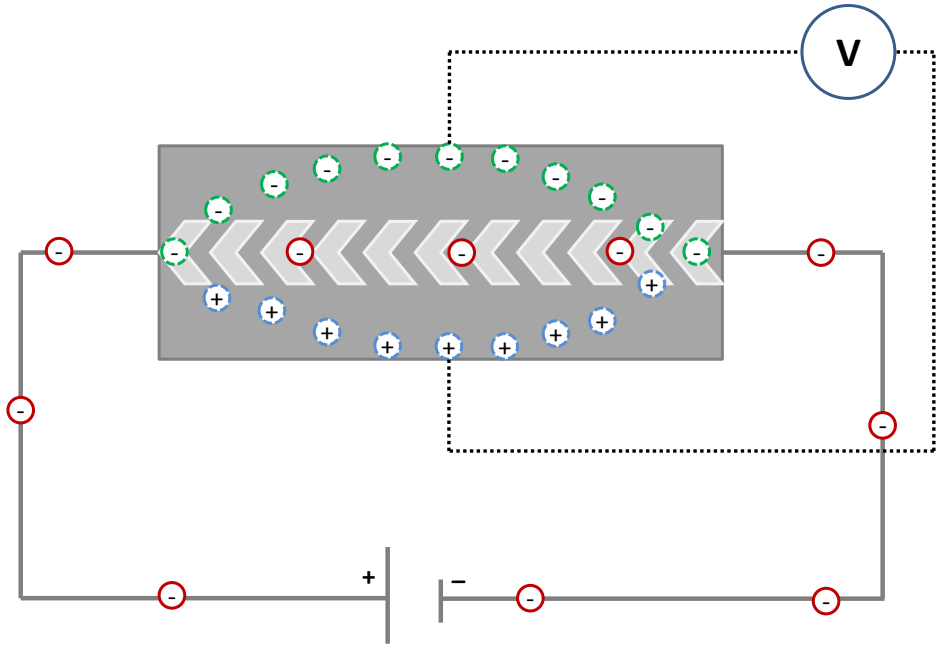


Abbildung 2.7: Der Stromfluss (rot) durch eine konduktive Platte wird durch ein magnetisches Feld, vertikal zur Platte, gestört (Feld nicht eingezeichnet). Der Elektronenfluss wird dadurch z.B. nach oben abgedrängt (grün) und unten entsteht ein Elektronenmangel (blau). Ein elektrisches Feld entsteht, dessen Spannung proportional zur Magnetfeldstärke vertikal zur Platte ist.

2.2.3 Magnetometer

Ein weiterer Sensor, der häufig in IMUs verbaut ist, ist das Magnetometer. Dieses hat den Zweck, magnetische Flussdichten, beziehungsweise Magnetfelder, zu messen. Damit lässt es sich zum Beispiel als magnetischer Kompass verwenden. Auch bei diesem Sensor liegt der Fokus auf der mikroelektromechanischen Ausführung, aus den gleichen Gründen, wie bei den anderen beiden Sensoren. MEMS-Magnetometer unterscheiden sich beim Messprinzip von den bisherigen Sensoren, da dieses magnetisch ist und nicht mehr kapazitiv. Dabei gibt es zwei dominierende Effekte zur Messung. Die Messung kann über den Hall-Effekt in nichtmagnetischen Materialien oder den magnetoresistiven Effekt in magnetischen oder hybriden Materialien erfolgen. Da aber fast 90% der im Handel erhältlichen Magnetometer den Hall-Effekt nutzen, wird dieser Abschnitt auch auf eben solche Magnetometer eingeschränkt [Ned15].

Zur Nutzung des Hall-Effekts wird eine konduktive Platte verwendet, durch welche Strom fließt. Dadurch bildet sich ein gerader Elektronenstrom von der einen Seite der Platte zur

anderen. Dieser Fluss wird gestört, sobald die Platte einem Magnetfeld ausgesetzt wird, welches orthogonal zur Stromrichtung verläuft. Dadurch werden die Elektronen auf die eine Seite der Platte gedrängt und auf der anderen Seite entsteht eine Elektronenmangel. Zwischen den gegenüberliegenden Seiten der Platte lässt sich anschließend eine Spannung messen, die von der Stärke und Richtung des Magnetfeldes abhängt. Es hat sich demnach ein elektrisches Feld gebildet, welches senkrecht zur Stromrichtung und zum Magnetfeld verläuft [Du14; Ned15]. In Abbildung 2.7 ist dieses Verhalten abgebildet. Über die Spannungsmessung lässt sich anschließend die Stärke einer Richtung des Magnetfelds errechnen [Ned15].

Wie auch bei den vorherigen beiden Sensoren werden drei einachsige Sensoren benötigt, um Messgrößen im dreidimensionalen Raum messen zu können. Dafür werden auch drei Magnetometer, beziehungsweise konduktive Platten orthogonal zueinander ausgerichtet, so dass diese den magnetischen Fluss in alle drei Richtungen messen können. Die Einheit der Messwerte ist Tesla und die Sensitivität ist fester Bestandteil der Komponenten, kann also nicht, wie bei den anderen Sensoren, durch Skalierung verändert werden [Du14].

Aktuelle Magnetometer nach diesem Prinzip arbeiten meist mit Aktualisierungsfrequenzen von 100Hz bis 160Hz [Du14; Hui+15]. Damit sind sie nur knapp 10% so schnell wie aktuelle mikroelektromechanische Accelerometer und Gyroskope. Dieser Frequenzunterschied ist bedingt durch das magnetische Messprinzip, welches dem kapazitiven Messprinzip in der Aktualisierungsgeschwindigkeit unterlegen ist. Nach einer Magnetfeldänderung muss eine gewisse Zeit gewartet werden, bis das messbare, orthogonale, elektrische Feld aufgebaut wurde und stabil anliegt [Du14]. Deshalb ist die Abtastfrequenz bedeutend geringer als die der anderen Sensoren. Bei 100Hz sind erwartet alle 10ms neue Werte abrufbar.

Die Nachteile beim Magnetometer sind für den hier gewünschten Anwendungsfall gravierend. Die geringere Frequenz kann dafür sorgen, dass die Daten der Magnetometer nicht genügend Aktualität aufweisen. Außerdem sind Magnetometer viel störungsanfälliger als kapazitive Sensoren, da bereits magnetische Felder von elektrischen Leitern die Messwerte beeinflussen können [Du14; Hui+15]. Die falschen Werte müssen demnach effizient erkannt und herausgefiltert werden.

Die Nutzung als magnetischer Kompass bringt für den geforderten Anwendungsfall keinen ersichtlichen Nutzen. Die Kombination der Messwerte mit denen der anderen Sensoren kann aber Vorteile bringen. Zum einen können Magnetometerdaten beispielsweise als Positionsreferenz genutzt werden, um anschließend mit Hilfe der Accelerometerdaten die exakte Position einer Hand zu bestimmen. Andererseits können die Daten als Referenz für die anderen Sensoren dienen, die durch Abgleiche ihren Genauigkeitsfehler über die Zeit reduzieren können. Alleine verspricht das Magnetometer keine großen Vorteile und ist aufgrund seiner Nachteile ein eher optionaler Kandidat, was die Sensorwahl für den gegebenen Ansatz angeht. In Kombination könnte es aber durchaus einige Vorteile mit sich bringen, weshalb, wenn möglich, die im Folgenden verwendete IMU ein Magnetometer besitzen sollte.

3 Anbindung externer Inertialsensoren

Die Auswahl passender Sensorik, sowie weiterer Hardwarebauteile zur Steuerung dieser, ist ein bedeutender Faktor im Rahmen der Entwicklung eines Prototypen. Der Prototyp hat die Aufgabe, die Sensorik auf der Hand der Anwender zu platzieren und dafür zu sorgen, dass diese möglichst zuverlässig, synchron, hochfrequent und fehlerlos Daten an ein Touchgerät übermitteln kann. Die letzteren drei Eigenschaften gelten dabei als Kriterien für die Auswahl von Komponenten. Für das Kriterium Hochfrequenz wird dabei ein Richtwert von etwa 500Hz veranlasst, also sollen mindestens alle 2ms aktuelle IMU-Daten erhalten werden. Dieser Wert orientiert sich an den gerade noch wahrnehmbaren Latenzunterschieden von 2,38ms, welche Ng et al. in ihrer Arbeit untersuchten [Ng+12]. Ebenso der Wert für die Synchronizität. Zwischen zwei IMU-Paketen unterschiedlicher IMUs dürfen maximal 2ms liegen, damit sie als synchron wahrgenommen werden und ihre Daten etwa gleich alt sind. Die Machbarkeit dieser Arbeit mit heutiger Hardware ist ein limitierender Faktor, den es zu beachten gilt, da sie die Wahl der benutzbaren Komponenten einschränkt. Es soll in Zukunft möglich sein, den hier umgesetzten Ansatz mit Hilfe von Smartwatches, Smartpens oder Smartrings nachzubilden. Aus diesem Grund muss der Prototyp auch die selbe Sensorik verwenden, wie die, der besagten Geräte. In Smartgeräten kommen meist 9DoF-IMUs zum Einsatz, welche im vorherigen Kapitel bereits dargestellt wurden. Die Auswahl geeigneter IMUs ist somit der Kernaspekt bei der Erstellung eines Hardware-Prototypen und ausschlaggebend für die Erfüllung der drei Kriterien:

- **Synchronizität** Die zeitliche Differenz zwischen Daten zweier IMUs darf höchstens 2ms betragen.
- **Hochfrequenz** Die Frequenz, mit welcher alle IMUs abgetastet werden können, muss etwa 500Hz oder mehr betragen.
- **Fehlerlosigkeit** Die Fehlerrate sollte so weit minimiert werden, wie dies unter Einhaltung der ersten zwei Kriterien möglich ist.

Die Schritte zur Erstellung des Prototypen, welcher in der ersten, hier durchgeführten Studie zum Aufzeichnen von IMU-Daten verwendet wurde und später die Eingabedaten der IMUs für das neuronale Netz lieferte, wird im Folgenden beschrieben. Dazu werden als Erstes alternative Designkonzepte und Komponenten vorgestellt und diskutiert, warum diese sich nicht für den hier realisierten Anwendungsfall eignen. Dazu wird Bezug auf die obigen Kriterien Synchronizität, Hochfrequenz und Fehlerlosigkeit genommen. Anschließend wird das finale, in dieser Arbeit verwendete Konzept vorgestellt und darauf eingegangen, warum es sich dabei um das geeignetste Design handelt. Abschließend wird erklärt, wie der Prototyp auf die

menschliche Hand übertragen wurde, um die einzelnen IMUs an den drei gewählten Positionen zu platzieren. Für alle getesteten Designs wurde als Touchgerät ein Asus Google Nexus 7 (2013)[1] mit Android-Betriebssystem verwendet, mit einer Bildschirmauflösung von 1920 x 1200 Pixeln und einer Touchscreen-Abtastrate von 60Hz bei einer Displaygröße von 7 Zoll.

3.1 Alternative Designkonzepte

Die hier aufgeführten Konzepte werden in derselben Reihenfolge beschrieben, in welcher sie auch getestet wurden. Sie bauen demnach aufeinander auf und stellen die Entwicklung des finalen Designs dar. Bei jedem Design werden die einzelnen Komponenten und ihre Verbindungen detailliert vorgestellt. Bereits in einem vorherigen Design beschriebene Komponenten werden lediglich wieder aufgegriffen, jedoch nicht erneut im Detail beschrieben.

3.1.1 Kabelloses Design

Zur kabellosen Realisierung eines Prototypen wurde anfangs ein „IMUduino“[2] ausgewählt, bei welchem es sich um einen Klon des Arduino Leonardo handelt. Der Arduino Leonardo ist ein häufig genutztes, programmierbares E/A-Board mit verbautem Mikrocontroller, welches es erlaubt, IMUs zu verbinden und zu steuern. Der IMUduino hat gegenüber seinem Vorbild einige Vorteile: Er besitzt eine bereits integrierte 10DoF-IMU und verfügt über einen „Bluetooth Low Energy“-Chip, bei nur etwa einem Viertel der Größe des Arduino Leonardo und dem gleichen Mikrocontroller.

Bei der verbauten IMU handelt es sich um eine MPU-6050[3] von InvenSense, die aus je einem dreiachsigen Accelerometer und Gyroskop besteht, in Kombination mit dem dreiachsigen Magnetometer HMC5883L von Honeywell[4]. Die Ausgabefrequenzen von Accelerometer und Gyroskop belaufen sich auf 1kHz [Mpua], was bedeutet, dass die Werte dieser Sensoren jede Millisekunde aktualisiert werden. Die Ausgabefrequenz des Magnetometers ist mit 160 Hz [Maga] ausreichend, um etwa alle 7ms neue Werte zu erhalten. Somit ist das Kriterium der Hochfrequenz für die 10DoF-IMU des IMUduino erfüllt, vorausgesetzt diese Frequenz sinkt nicht, beim Übertragen der Daten an das Touchgerät oder beim Verbinden weiterer IMUs. Der IMUduino verfügt über ausreichende Sensorik, um die übliche Sensorik eines Smartgeräts realistisch zu imitieren. Bei dem zehnten Freiheitsgrad der IMU handelt es sich um ein einachsiges Barometer, welches aber in diesem Design ungenutzt bleibt. Auch die Größe

[1] Spezifikationen: http://www.gsmarena.com/asus_google_nexus_7_(2013)-5600.php
[2] Kickstarter Projekt: https://www.kickstarter.com/projects/1265095814/imuduino-wireless-3d-motion-html-js-apps-arduino-p
[3] Herstellerwebsite: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/
[4] Spezifikationen: http://www.digikey.com/product-detail/en/honeywell-microelectronics-precision-sensors/HMC5883L-TR/342-1082-1-ND/2507853

des kompletten Chips mit etwa $40\,mm \times 16\,mm$ entspricht völlig den Kriterien, da diese Größe dafür geeignet ist, auf Finger, Stylus und Handgelenk angebracht zu werden.

Die Idee dieses Konzepts besteht also darin, mehrere IMUduinos mit kleinen Akkus auszustatten und auf einer Hand oder einem Stylus zu platzieren, an den drei Stellen, an denen auch ein Smartgerät sitzen würde (Finger, handgelenk, Stylus). Die Werte sollen dann per Bluetooth an das ausgewählte Touchgerät übertragen werden, auf welchem die Latenz später reduziert werden soll. Der im IMUduino verbaute Bluetooth Low Energy v4.0 Chip „Nordic nRF8001“[5] kann 20 Byte Daten pro Paket per Bluetooth übermitteln [Btl]. Die Rohwerte jedes IMU-Sensors jeder Achse bestehen aus 2 Byte. Sie lassen sich somit in einem Bluetooth-Paket zusammenfassen und verschicken, da sie gebündelt nur 18 Byte groß sind.

Auf der Android-Seite wird die Kommunikation über einen Bluetooth-Service gesteuert. Auf dem Android-Gerät wird dazu ein Bluetooth „Generic Attribute Profile“ (Abkürzung: Gatt) Server gestartet. Dieser erkennt andere Bluetooth-Geräte und kann den Zweck von Bluetooth-Paketen interpretieren, zum Beispiel einen Verbindungsversuch, Datenübertragungen oder Trennanfragen. Ist die Verbindung zu einem Gerät, wie zum Beispiel dem IMUduino, hergestellt, wird bei jedem Bluetooth-Paket ein „Callback“ auf dem Android-Gerät aufgerufen, das Paket richtig interpretiert und im Falle einer Datenübertragung werden die Daten angenommen.

In den Tests mit nur einem IMUduino konnte per Bluetooth nur etwa alle 40ms ein Paket mit den IMU-Daten versendet werden, was nur einer Frequenz von 25Hz entspricht. Die Paketumlaufzeit (auch Rundreisezeit, englisch: Round Trip Time, Abkürzung: RTT), also die Zeit zwischen Senden und Erhalt der Annahmebestätigung durch das andere Gerät, mit einem Google Nexus 7 Tablet über 1000 Pakete gemessen, beträgt durchschnittlich 41,57ms bei einer Standardabweichung (Abkürzung: SD) von 4,93ms.

Bei Betrachtung der zum Chip gehörigen Bibliothek „Adafruit_BLE_UART.cpp“[6] fällt auf, dass in jedem Bluetooth-Schreib-Befehl eine gewollte Verzögerung von 35ms fest einprogrammiert wurde. Diese ist sogar mit dem Kommentar „benötigt“ gekennzeichnet worden. Durch Abändern und Testen mit verschiedenen Verzögerungszeiten, an dieser Stelle der Bibliothek, kann die Sendefrequenz stark verbessert werden. Verzögerungszeiten von weniger als 20ms heben die Bluetooth-Sendefrequenz des IMUduino auf das Doppelte, also etwa ein Paket alle 20ms (Durchschnitt über 100 Pakete: 21,43ms, SD = 2,46ms), führen aber nach wenigen Paketen zum Zusammenbruch der Bluetooth-Verbindung mit dem anderen Gerät.

In den hier durchgeführten Tests wurden die besten Ergebnisse mit Verzögerungen von 24ms erzielt. Die Verbindung blieb dabei mehrere Minuten aktiv und stabil, bei einer erhöhten Sendefrequenz von etwa einem Paket alle 30ms, umgerechnet etwa 33Hz. Die Paketumlaufzeit verbesserte sich dadurch auf durchschnittlich 30,87ms (SD = 5,83ms, gemessen über 1000 Pakete). Trotzdem traten Verbindungsabbrüche signifikant häufiger auf, als mit der Standardverzögerung von 35ms. Verzögerungen zwischen 24ms und 30ms erzielten keine merklich

[5] Herstellerwebsite: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF8001
[6] Bibliothek zum Bluetooth Chip: https://github.com/adafruit/Adafruit_nRF8001

stabilere Verbindung als 24ms und Verzögerungen zwischen 31ms und 34ms erzielten keine erheblich höheren Übertragungsfrequenzen.

Zu erklären ist diese Verzögerung durch die Betrachtung des Datenblatts des Chips. Zwei Faktoren sind hier ausschlaggebend: Das „Active Signal“ und die Länge des Verbindungsintervalls. Das „Active Signal“ ist ein Informationssignal, welches das verbundene Gerät benachrichtigt, wann eine Verbindung stattfindet und Daten übertragen werden. Ohne dieses Signal können also keine Daten übertragen werden, da die lesende Anwendung nicht weiß, wann die Daten anliegen und so zwischen mehreren Schreibvorgängen unzusammenhängende Bits lesen würde. Das Verbindungsintervall beschreibt die Zeit, wie lange eine Verbindung bestehen muss, um ein oder mehrere Pakete übertragen zu können. Im Rahmen der Tests für den vorliegenden Bluetooth Chip ließ sich beobachten, dass pro Verbindungsintervall, ohne Ausnahme, nur ein Paket übertragen wurde. Beträgt nun das Verbindungsintervall zwischen den Geräten weniger als 30ms, so wird das „Active Signal“ automatisch deaktiviert und die Verbindung bricht ab [Btl].

Leider ist diese hohe Verzögerung bei derzeitigen Bluetooth Low Energy Chips normal, da durch eine rapide Senkung der Übertragungsfrequenz Energie gespart werden kann. Für dieses Konzept wäre eine Nutzung von normalen Bluetooth Chips denkbar, was aber aus mehreren Gründen wenig Sinn macht, obwohl dann Frequenzen von 1kHz durchaus möglich wären. Zum einen ist der Energiebedarf deutlich höher, wodurch sich der benötigte Akku in kurzer Zeit entladen würde. Zum anderen werden in Smartgeräten, zu welchen ein möglichst naher Bezug existieren soll, fast ausschließlich Low Energy Chips verbaut, da diese stromsparend und klein sind.

Zusammenfassend ist bei diesem Designkonzept festzuhalten, dass das Kriterium der Hochfrequenz aufgrund der limitierten Bluetooth-Übertragungsfrequenz für den Prototypen nicht eingehalten werden kann. Die Erhöhung der Übertragungsfrequenz ist ebenfalls keine valide Option, da sie die Stabilität und Zuverlässigkeit gefährdet und somit das Kriterium der Fehlerlosigkeit an den Prototypen verletzt.

3.1.2 Arduino-Verbund

Bis auf die Bluetooth-Übertragung versprechen die Komponenten des ersten Ansatzes gute Ergebnisse. Eine weitere Option besteht nun darin, die Bluetooth-Verbindung durch eine USB-Verbindung zu ersetzen. Über einen USB On-The-Go (kurz: OTG) Adapter an der USB-Schnittstelle des Android-Touchgerätes kann der IMUduino verbunden werden. Ein „USB-Broadcastreceiver“ auf der Android-Seite erkennt den Zweck jedes USB-Pakets und ruft die passende, definierte Callback-Methode für jedes Paket von jedem verbundenen Gerät auf, in welcher zum Beispiel Daten, die am USB-Port anliegen, angenommen werden können. Eine 9DoF-IMU eines IMUduinos ist allerdings nicht ausreichend, da für diese Arbeit drei 9DoF-IMUs genutzt werden sollen, um den Einfluss der drei gewählten Platzierungen aus Kapitel 1 zu untersuchen.

Abbildung 3.1: Gezeigt wird der Anschluss von einem IMUduino (unten), einem Arduino Micro (mittig) und einem Arduino Nano (oben) über einen USB-OTG Hub an das Google Nexus 7 Tablet.

Zwei andere Arduinos wurden deshalb hinzugezogen. Dabei handelte es sich um einen Arduino Nano und einen Arduino Micro, welche beide keine eigene IMU besitzen, aber leicht mit einer externen IMU verbunden werden können. Die drei Arduinos werden bei diesem Konzept über einen USB-OTG Hub mit dem Android-Gerät verbunden, wie in Abbildung 3.1 gezeigt.

Der USB-Broadcastreceiver auf der Android-Seite wird dementsprechend angepasst, dass dieser die Geräte-ID der verbundenen Arduinos ausliest und je nach Herkunft eines Datenpakets eine andere Callback-Methode aufruft, um die Daten zu verarbeiten. Bei einer Verwendung von drei gleichen Arduinos mit der gleichen ID wäre es zur eindeutigen Differenzierung nötig, beispielsweise den Gerätenamen über Treibermodifikation oder das Flashen eines umbenannten Bootloaders mit einem Zusatz zu versehen. Über den Namenszusatz ließe sich das Gerät dann identifizieren.

Die Kommunikation über USB bringt einige Vorteile mit sich. Als erstes ist hier die Geschwindigkeit zu nennen. Der IMUduino und der Arduino Micro schaffen es, ohne Modifikation, bei der Übertragung von 20 Byte Paketen eine USB-Übertragungsfrequenz von über 1kHz zu erreichen. Damit ist es möglich, mehr als ein 20 Byte Paket pro Millisekunde zu versenden. Ebenfalls ist es möglich, mehr als 20 Byte in einem Paket zu versenden.

Praktisch arbeiten die wenigsten IMUs mit Aktualisierungsfrequenzen über 1kHz, was demnach der limitierende Faktor der Übertragung ist. Auch das bisher genutzte IMUduino kann die Werte seiner IMUs nur mit maximal 1kHz abrufen und über USB senden, im Falle des Magnetometers sogar nur mit 160Hz, obwohl der verbaute USB- und Mikrocontroller viel höhere Übertragungsraten für diese geringen Bytegrößen unterstützen würde [Atmb]. Der gleiche Mikrocontroller, mit USB-Controller, mit der Bezeichnung ATmega32U4 [Atmb], kommt auch

im Arduino Micro zum Einsatz. Des Weiteren ist die Größe der Pakete nicht mehr auf 20 Byte Daten beschränkt, sondern ein Paket kann bis zu 63 Byte Daten enthalten. Zu nennen ist noch, dass die Kommunikation über USB zuverlässiger funktioniert, als eine Kommunikation über Bluetooth, bei der bereits andere Bluetooth-Verbindungen im Raum zu Fehlern bei der Übertragung führen können [EH01]. Im Arduino Nano kommt zwar ein anderer Mikrocontroller zum Einsatz, der aber ähnlich gute Werte aufweist, nämlich der ATmega328 [Atma].

Das Problem beim Arduino Nano ist allerdings, dass der Mikrocontroller keinen zusätzlichen USB-Controller besitzt, wie der ATmega32U4. Dadurch wird eine Kommunikation über USB sehr umständlich und langsam, da sie emuliert werden müsste. Der genannte Arduino eignet sich folglich nicht für diesen Aufbau. Abgesehen davon ist das Hauptproblem bei einem Verbund von Arduinos ein anderes.

Sobald die Arduinos die Werte ihrer IMUs gelesen haben, schreiben sie diese an ihre USB-Schnittstelle. Sobald sich an einer USB-Schnittstelle Werte befinden, liest das Android-Gerät diese Werte, wodurch es passieren kann, dass beispielsweise von einem Arduino drei Mal hintereinander Werte gelesen werden, während von einem zweiten keine Werte gelesen werden. Durch die ungleiche Anzahl von Lesevorgängen würden nun eventuell bestimmte IMUs, aufgrund von mehr ausgelesenen Werten, bessere Ergebnisse im späteren maschinellen Lernen erzielen. Dadurch wären keine Vergleiche mit den IMUs auf den anderen zwei Positionen mehr möglich. Unter den Arduinos besteht keine Kommunikation, weshalb das Android-Gerät die Arduinos synchronisieren muss. Das kann über Warteaktionen geregelt werden, wodurch manche IMU-Werte allerdings länger auf die Verarbeitung warten müssen, als unbedingt nötig.

Alles in allem ist dieser Aufbau nicht ideal. Gewinnbringend ist aber die Erkenntnis, dass die USB-Übertragungseigenschaften der zwei ATmega32U4 Arduinos ideale Voraussetzungen für eine hochfrequente und fehlerlose USB-Kommunikation mit sich bringen. Lediglich die fehlende Synchronizität ist ein Problem, welches sich über einen Verbund mit Hilfe eines USB-OTG Hubs nur schwer lösen lässt.

3.1.3 Zwischendesigns

Die vorherigen Designs haben gezeigt, dass eine USB-Kommunikation für eine möglichst zuverlässige, hochfrequente und fehlerfreie Datenübertragung eindeutig einer Bluetooth Low Energy v4.0 Lösung vorzuziehen ist. Um allerdings gleiche Bedingungen für alle drei IMUs zu schaffen, die später auf der Hand und einem Stylus platziert werden sollen, ist es notwendig, drei gleiche IMUs zu verwenden, um später bessere Ergebnisse aufgrund von besseren IMUs ausschließen zu können.

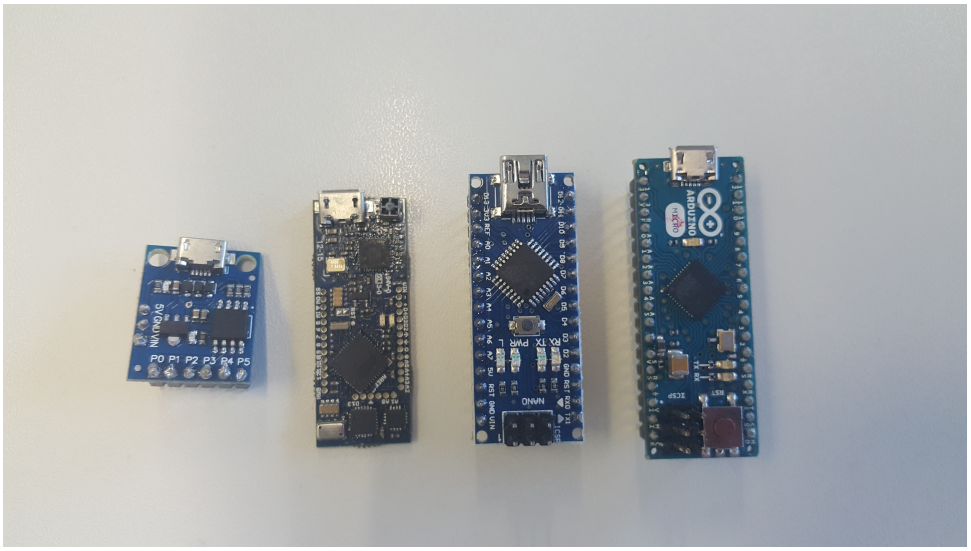


Abbildung 3.2: Von links nach rechts sind ein ATtiny85, ein IMUduino, ein Arduino Nano und ein Arduino Micro zu sehen.

Gute Ergebnisse verspricht die 9DoF-IMU MPU-9250 [Mpub][7]. Diese besteht aus einer MPU-6500, also einer verbesserten Version der MPU-6050 aus dem IMUduino, und ist in der hier verwendeten Ausführung mit einem dreiachsigen AK8963 [Magb] Magnetometer gepaart. Die MPU-6500 besteht, wie die MPU-6050, aus einem dreiachsigen Accelerometer und einem dreiachsigen Gyroskop. Die Aktualisierungsfrequenz des Gyroskops beträgt 1kHz und die des Accelerometers sogar 4kHz, es ist also möglich, von diesen beiden Sensoren jede Millisekunde neue Werte zu bekommen [Mpub]. Das Magnetometer ist mit 100Hz Aktualisierungsrate etwas langsamer als das des IMUduinos, was dazu führt, dass sich das Ausgangssignal nur alle 10ms ändert, statt etwa alle 7ms, wie beim IMUduino [Magb]. Weiterhin unterstützt die genannte IMU sowohl eine I²C-Datenverbindung (Inter-Integrated Circuit) [Phi], als auch eine SPI-Verbindung (Serial Peripheral Interface) [Büc06] zur Kommunikation mit Mikrocontrollern, was Flexibilität ermöglicht.

Aufgrund der guten Eigenschaften und der Einheitlichkeit werden also in den folgenden Designs nurnoch drei MPU-9250 IMUs verwendet. Als Mikrocontroller, zur Steuerung der IMUs und zur Kommunikation mit dem Touchgerät, soll aufgrund seiner geringen Größe ein ATtiny85 verwendet werden [Att]. In Abbildung 3.2 ist ein Größenvergleich der bisher genutzten Mikrocontroller mit dem ATtiny85 zu sehen.

Das Problem dabei ist jedoch, wie beim Arduino Nano, dass der ATtiny85 keinen eigenen USB-Controller besitzt. Eine Emulation der USB-Schnittstelle ist aber mit Hilfe der „DigiCDC“[8] Bibliothek möglich, welche eine virtuelle, serielle Schnittstelle an der USB-Schnittstelle erzeugt.

[7] Herstellerwebsite: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/
[8] Bibliothek für USB-Schnittstelle: https://digistump.com/wiki/digispark/tutorials/digicdc

Darüber können anschließend die Daten ausgetauscht werden. Hierbei entsteht der Nachteil, dass die Kommunikation über die virtuelle Schnittstelle, aufgrund des fehlenden Hardware USB-Controllers, extrem langsam verläuft. Tests mit diesem Design konnten gerade einmal ein Byte pro Millisekunde (Durchschnitt = 1,01ms pro Byte, SD = 0,52ms pro Byte, gemessen über 1000 Byte) übertragen.

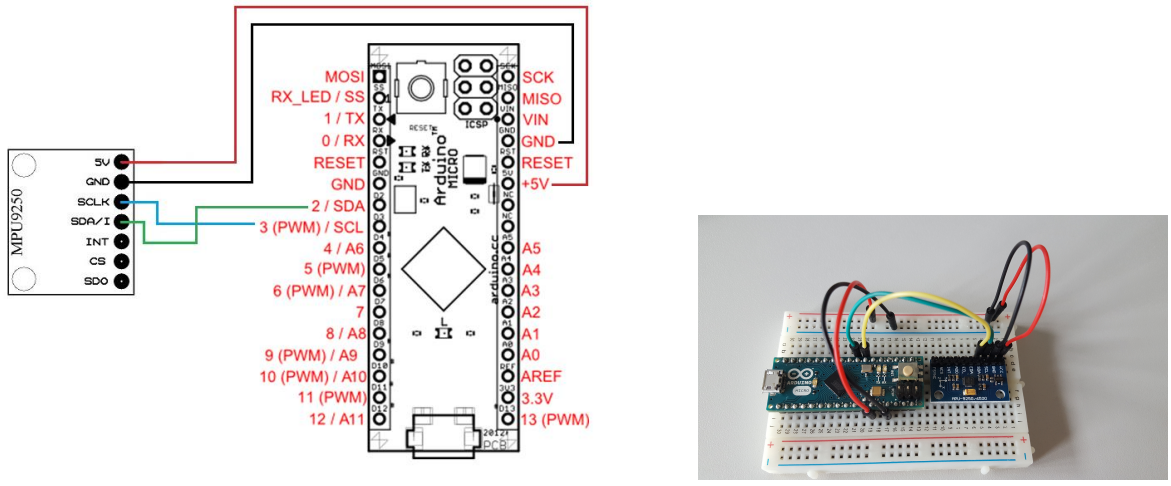
Da dies zu langsam ist, um jede Millisekunde die Daten von drei IMUs zu übertragen, wird der ATtiny85 durch einen anderen Mikrocontroller ersetzt. In den vorherigen Tests hat der Arduino Micro gute USB-Kommunikationseigenschaften für den hier geforderten Anwendungsfall bewiesen. Die IMUs sollen demnach über den Mikrocontroller des Arduino Micro gesteuert werden und die Werte, welche der Arduino von den IMUs abfragt, sollen von ihm über USB an das Android-Gerät übermittelt werden. Für die USB-Verbindung wird wieder ein USB-OTG Adapter verwendet.

Um dieses Design zu verwirklichen, muss nur noch die Kommunikation zwischen den IMUs und dem Arduino implementiert werden. Naheliegend ist eine Kommunikation mit Hilfe des I²C-Datenbuses. Dieser ist beim Lesen der IMU-Register, welche die Sensorwerte enthalten, mit 400kHz pro Register [Mpub], ausreichend schnell. Es müssen 18 Register hintereinander gelesen werden und bei einer Abfragerate von 400kHz pro Register bleibt die Gesamtabfragefrequenz für alle Register über 1kHz, was vorteilhaft für dieses Design ist. Vorteilhaft ist auch die einfache Anbindung von IMUs über I²C, da nur zwei Verbindungen benötigt werden. Diese sind eine serielle, bidirektionale Datenverbindung (SDA) und ein serielles Clock-Signal (SCL) als Taktgeber für die IMUs [Phi], wie in Abbildung 3.3[9] zu erkennen.

Außerdem bietet I²C viele Methoden zur einfachen Implementierung des Mikrocontroller-Programms. Das ausgewählte Arduino Micro bietet je einen SDA- und einen SCL-Pin, über welche alle drei MPU9250-IMUs angeschlossen werden sollen. Die IMUs fungieren dann als „Slaves“ des Arduinos, welcher als „Master“ agiert und die Kommunikation über das Setzen eines „R/W-Bits“ in den Bitadressen der IMUs kontrolliert. Setzt er dieses Bit auf 1, so sendet die IMU Sensordaten an den Arduino (Zustand R für Lesen (englisch: Read)), hat das Bit den Wert 0, so wartet die IMU auf Signale des Arduinos (Zustand W für Schreiben (englisch: Write)). Das Anschließen aller drei IMUs ist also möglich, wenn zu jedem Zeitpunkt nur eine IMU das entsprechende Bit auf 1 gesetzt hat und somit Daten sendet. Danach wird sie, durch Setzen des Bits auf 0, deaktiviert und die nächste IMU wird aktiviert.

Das Durchrotieren der einzelnen IMUs beansprucht aufgrund des Umstellens der IMUs durch Bitwechsel zusätzliche Zeit. In hier durchgeführten Tests konnte im Schnitt alle 5ms ein Paket einer IMU empfangen werden. Die exakte Paketumlaufzeit war damit 4,73ms (SD = 1,38ms, gemessen über 1000 Pakete) für die Daten einer IMU. Diese Frequenz wäre zwar akzeptabel, jedoch führt das zu dem Problem, dass die empfangenen Daten der einzelnen IMUs bis zu 10ms auseinander liegen für eine Messung. Möchte man diese IMU-Daten später mit anderen Daten

[9] Layout der IMU: https://drotek.com/shop/en/drotek-parts/421-mpu9250-gyro-accelerometer-magnetometer.html, Layout des Arduino Micro: https://www.arduino.cc/en/uploads/Main/ArduinoMicro_Pinout3.png



(a) Layout **(b)** Aufbau

Abbildung 3.3: Dargestellt sind Abbildungen der I²C-Verbindung zwischen IMU und Arduino, sowohl als Layout (a), als auch als Foto eines hier konstruierten Aufbaus (b).

verknüpfen, so könnte eine IMU bessere Ergebnisse erzielen, allein aufgrund der Tatsache, dass sie früher abgefragt wurde und damit die Zeitpunkte der IMU-Daten näher an den Zeitpunkten der anderen Daten liegen. Damit ist das Kriterium der Synchronizität verletzt. I²C eignet sich abschließend nur bedingt als Datenbus für den geforderten Anwendungsfall, weshalb im finalen Design ein anderer Datenbus genutzt wird.

3.2 Finales System Design

Das finale Design des Systems verwendet weiterhin dieselben Komponenten, die bereits im vorherigen Design genutzt wurden, da sie sich durch gute Abfrage- und Sendegeschwindigkeiten als geeignet herausgestellt haben. Als Mikrocontroller-Board kommt das Arduino Micro zum Einsatz und als IMUs drei MPU9250. Beim letzten Design war der I²C-Bus aufgrund erhöhter Abfragezeiten und entstehender Asynchronizität ein Problem, welches in diesem Design durch einen anderen Datenbus behoben werden soll. Beide Komponenten unterstützen neben I²C auch das „Serial Peripheral Interface“ (kurz SPI) [Büc06].

Zur Nutzung dieser Schnittstelle sind mehr Verbindungen zwischen den einzelnen IMUs und dem Arduino erforderlich, als bei einer I²C-Verbindung. Statt einer einzelnen bidirektionalen Datenverbindung sind nun zwei von Nöten. Eine für die Kommunikation vom „Master“ zu den „Slaves“ (MOSI; Master Output Slave Input) und eine für die Kommunikation in die andere Richtung (MISO; Master Input Slave Output). Weiterhin wird ebenfalls ein Clock-Signal benötigt, für die Synchronisation der einzelnen IMU-Slaves. Zusätzlich wird ein „Slave-Select“-Signal pro IMU benötigt, welches bestimmt, welche IMU (Slave) gerade mit dem Arduino (Master) kommunizieren darf. Das Layout der Verbindungen ist in Abbildung 3.4

dargestellt. Die Beschriftungen und Anzahl der Pins der IMU-Chips unterscheiden sich je nach Ausführung [Mpub]. Die hier verwendeten Chips lassen sich in Abbildung 3.5 betrachten, in welcher der für diese Arbeit genutzte Aufbau zu sehen ist.

Im Prinzip ähnelt diese Art von Kommunikation sehr der des I²C-Datenbuses, da hier ebenfalls durch die einzelnen Slaves sequentiell durchrotiert werden muss, aber SPI bietet einige Vorteile. So bietet beispielsweise der Mikrocontroller des Arduinos einen SPI-Modus mit doppelter Geschwindigkeit [Atmb] und alle Register der IMUs lassen sich statt mit 400kHz mit 1MHz abfragen [Mpub].

Diese Vorteile kommen aufgrund von weniger umfangreichen Kommunikationsvorgaben und der zwei unidirektionalen Datenleitungen, welche synchron genutzt werden können, statt der bidirektionalen Datenleitung bei I²C, welche für die Kommunikation in beide Richtungen geteilt werden muss. Die weniger umfangreichen Kommunikationsvorgaben machen es dafür notwendig, viele Einstellungen selbst vorzunehmen, welche bei der Nutzung des I²C-Buses vordefiniert sind. Essentiell ist hierbei die manuelle Konfiguration der einzelnen Pins des Arduinos als Output- und Input-Pins, sowie die Definitionen der einzelnen IMUs und ihrer zugehörigen Verbindungen im Arduino-Code.

Der wichtigste Vorteil liegt jedoch bei den IMUs selbst, da diese über einen „high speed SPI Mode“ verfügen, welcher nur noch Lesezugriffe auf die Sensorregister der IMUs zulässt, dafür

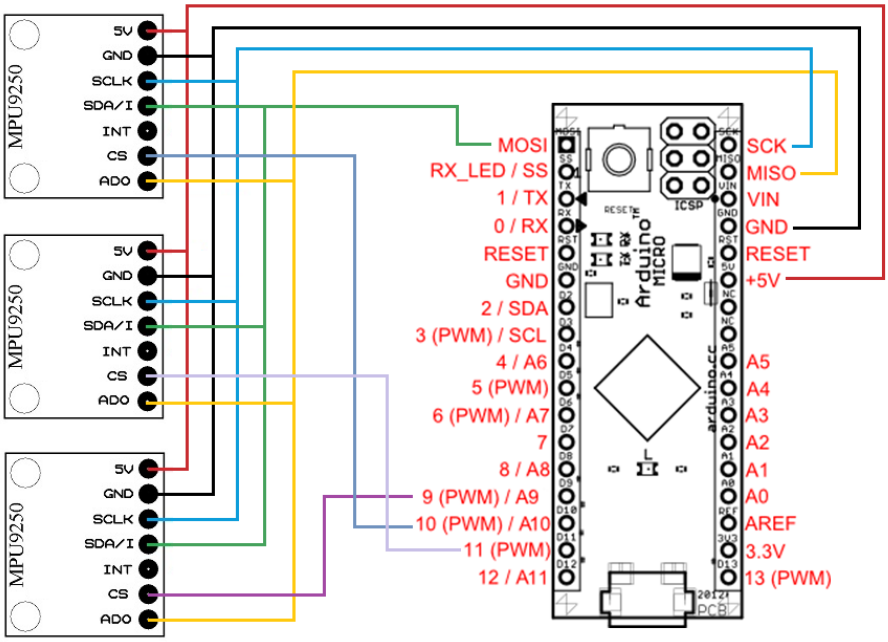


Abbildung 3.4: Dargestellt ist das Layout der SPI-Verbindungen zwischen den IMUs und dem Arduino Micro. Die Pinbezeichnungen können sich je nach Chipausführung bei den einzelnen Komponenten unterscheiden.

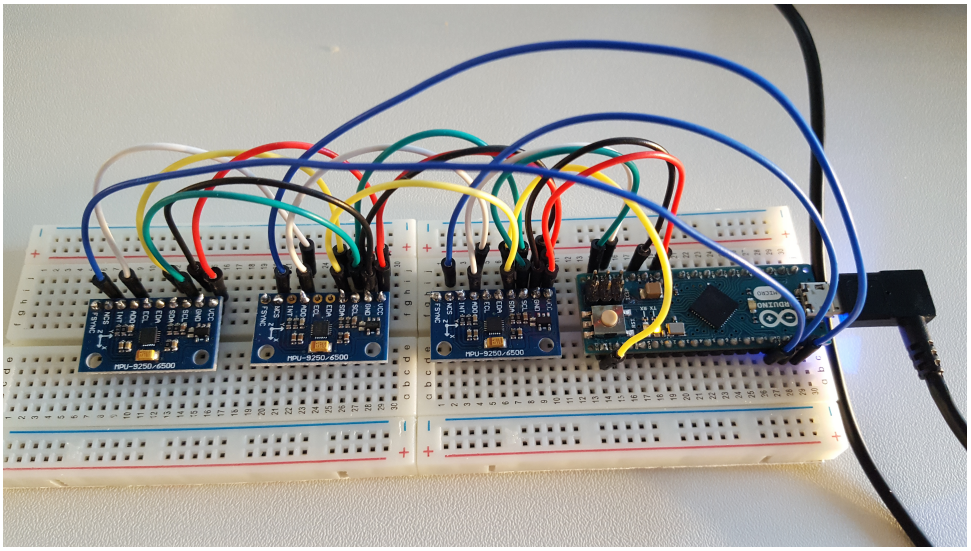


Abbildung 3.5: Dargestellt ist der Aufbau und die Verbindung über SPI der einzelnen, verwendeten Komponenten des finalen Designs. Die Verbindung der MPU9250s mit dem Arduino Micro entspricht dem Layout in Abbildung 3.4.

aber jedes Register mit 20MHz abgefragt werden kann [Mpub]. Durch diesen Modus aktualisieren sich alle Sensoren der IMUs kontinuierlich mit 1kHz außer das Magnetometer, welches seine Daten im kontinuierlichen Modus mit 100Hz aktualisiert [Magb]. Das Arduino muss in einer Dauerschleife die „Slave-Select“ Pins der einzelnen IMUs aktivieren, die Sensorregister auslesen und sobald von allen drei IMUs Daten erhalten wurden, muss es diese noch als Paket an das zugehörige Android-Gerät senden.

Ein Paket besteht demnach aus drei mal 18 Byte, da pro IMU 9×2 Byte Daten gelesen werden, womit eine Gesamtgröße von 54 Byte pro Paket aller drei IMUs zusammenkommt. Da dies unter der zulässigen Gesamtgröße von 63 Byte pro USB-Paket für den gewählten Mikrocontroller liegt [Atmb], wird die USB-Übertragungsgeschwindigkeit nicht reduziert. Mit diesem Prototyp durchgeführte Tests, im Rahmen dieser Arbeit, ergaben, dass im Schnitt das Abfragen der Werte einer IMU 300μs (Durchschnitt = 267,93μs, SD = 60,23μs, gemessen über 1000 Abfragen) benötigt. Das Abfragen aller drei IMUs zuzüglich dem Senden per USB als einzelnes Paket kommt damit durchschnittlich auf eine Millisekunde (exakter Durchschnitt: 0,96ms, SD = 0,36ms, bei 1000 Paketen, gesendet an ein Notebook).

Theoretisch ist also das Empfangen aller IMU-Werte auf dem Android-Gerät mit einer Frequenz von etwa 1kHz möglich. Außerdem erfüllt der hier beschriebene Prototyp alle Kriterien, die an ihn gestellt wurden. Die Daten der einzelnen IMUs liegen höchstens 600μs auseinander, was für den hier geforderten Anwendungsfall ausreichend synchron ist. Die Daten kommen mit etwa 1kHz hochfrequent an dem Gerät an und auch die Übertragung per USB ist fehlerfrei und zuverlässig. Ein Manko ist jedoch die Rate, mit welcher das Android-Gerät die Daten annimmt.

Das Arduino schreibt die Sensordaten in einen Buffer an der USB-Schnittstelle, welcher erst weiter beschrieben werden kann, wenn die bisherigen Daten daraus abgerufen wurden, wie in der zugehörigen Bibliothek[10] zu sehen ist. Die endgültige Frequenz, mit welcher alle IMU-Daten gebündelt zum Android-Gerät gelangen, ist also bei diesem Konzept nur noch von der Rate abhängig, mit welcher das Android-Gerät die Daten an der USB-Schnittstelle abruft. Ansonsten beträgt diese Frequenz 1kHz, wobei die Magnetometerdaten sich nur bei etwa jedem zehnten USB-Paket ändern, da dessen Aktualisierungsfrequenz mit 100Hz um das Zehnfache niedriger ist, als die der anderen Sensoren.

3.2.1 Software-Schnittstelle

Der Arduino des beschriebenen Designs sendet die Messdaten der IMUs als Daten-Pakete zu je 63 Byte. Zum Empfangen dieser Pakete wird auf der anderen Seite der USB-Verbindung ein „USB-BroadcastReceiver“ benötigt. Dieser erkennt angeschlossene USB-Geräte an der seriellen Schnittstelle des Android-Geräts und stellt eine Verbindung zwischen den Geräten her. Dafür interpretiert er den Zweck von USB-Paketen und erkennt, ob es sich dabei um Verbindungsversuche, Datenpakete oder andere Anfragen handelt. Kommt eine Verbindung zustande, dann definiert er grundlegende Eigenschaften der Kommunikation, wie etwa den seriellen Port mit entsprechenden Baud-Raten, Datenflusskontrolle, Stop- und Paritätsbits und mehr. Als serielle USB-Host Bibliothek wurde ein open-source Android USB-Controller Projekt verwendet[11].

Pro Gerät lässt sich anschließend am entsprechenden seriellen Port ein „USB-Callback“ definieren, welcher jedes Mal aufgerufen wird, wenn Daten vom verbundenen USB-Gerät gesendet werden. Hier werden dann die Datenpakete angenommen und anschließend der Buffer am Arduino geleert, damit dieser neue Daten senden kann. Zuerst wird überprüft, ob jedes Paket exakt 63 Byte groß ist, falls nicht, wird es verworfen, da es unvollständig ist, ansonsten angenommen. Anschließend müssen die angenommenen Sensorwerte wieder korrekt zusammengesetzt werden, da jeder Messwert zwei Byte groß ist, welche sich in zwei verschiedenen Registern der IMU befanden. Für jeden Messwert wird dazu zuerst das höherwertige Byte um 8 Bits nach links verschoben, wodurch ein 16 Bit „Short“ entsteht, dessen 8 niederwertige Bits alle den Wert 0 haben. Durch eine bitweise „Oder“-Operation mit dem niederwertigen Byte des übertragenen Messwertes werden diese Bytes schließlich auch mit den richtigen Bitwerten gefüllt. Wenn dieses Verfahren für alle 9 Messwerte aller drei inertialen Messeinheiten durchgeführt wurde, so sind alle 27 Messwerte auf dem Android-Gerät als Shorts verfügbar.

Dieser ganze Prozess wird auf dem Android-Gerät in einem separaten Thread ausgeführt, damit die Messdaten zu jeder Zeit am Gerät ankommen, egal, was darauf gerade ausgeführt

[10] Bibliothek für serielle Kommunikation für Arduinos: https://github.com/arduino/Arduino/blob/master/hardware/arduino/avr/cores/arduino/HardwareSerial.cpp

[11] Android USB-Controller Bibliothek: https://github.com/felHR85/UsbSerial

wird. Wird der Thread pausiert oder ausgelagert, zum Beispiel aufgrund erhöhter Auslastung des Gerätes durch Hintergrundprozesse, so werden über einen kurzen Zeitraum keine Daten angenommen, weshalb der Thread immer die höchste Priorität erhalten sollte.

Mit dem oben beschriebenen Design und einem Google Nexus 7 als Android-Gerät, mit geschilderter Software-Schnittstelle, beträgt die Zeit zwischen zwei IMU-Paketen des Arduinos durchschnittlich 1,66ms (SD = 0,79ms, gemessen über 1000 Pakete). Dies entspricht etwa einer Frequenz von 602,4Hz. Da jedes Paket vom Tablet bestätigt wird und der Arduino erst nach einer Bestätigung ein neues Paket zusammenstellt, entsprechen die 1,66ms der Paketumlaufzeit. Außerdem ist die Paketumlaufzeit von der Auslastung des Tablets abhängig, kann also bei verschiedenen Android-Geräten anders ausfallen. Der Unterschied bei der hier genannten Paketumlaufzeit, im Gegensatz zu den Paketumlaufzeiten der vorherigen Designs, besteht darin, dass ein Paket die Daten aller drei IMUs enthält und nicht nur die Daten einer IMU, wie zuvor. Damit sind alle anfangs definierten Kriterien zur Anbindung der externen Inertialsensoren erfüllt und der Prototyp muss nur noch auf den Anwender übertragen werden, um zum Einsatz zu kommen.

3.2.2 Übertragung auf den Anwender

Die inertialen Messeinheiten sollen an den verschiedenen Positionen platziert werden, welche in Kapitel 1 gewählt wurden, um die IMUs anderer Geräte simulieren zu können. Eine IMU soll sich am Handgelenk des Nutzers befinden, um eine Smartwatch zu imitieren, eine am Zeigefinger, zur Simulation eines Smartrings und die dritte an einem Touchstylus, um den Effekt einer IMU auf einem Stylus untersuchen zu können. Da alle drei IMUs von dem Arduino Micro verwaltet werden, bildet dieser eine zentrale Einheit.

Jede IMU wird über ein 40cm langes, 10-poliges Flachbandkabel mit dem Arduino verbunden, wie in Abbildung 3.6 zu erkennen. Der Arduino versorgt alle IMUs über einen 5V- und einen Erdungs-/Massepin mit Strom. Die MOSI (SDA Pin an der IMU), MISO (AD0 Pin an der IMU) und SCL Verbindungen nutzen bei allen IMUs dieselben Pins am Arduino, wie in Abbildung 3.4 zu sehen ist. Es bietet sich also an, den Arduino auf einem Steckbrett anzubringen, damit seine Pins über die internen Kanäle des Brettes mit allen IMUs verbunden werden können. Lediglich die „Slave-Select“ Verbindung benötigt für jede IMU einen anderen Pin am Arduino, damit dieser jede IMU einzeln ansteuern kann, wie in Abbildung 3.4 und 3.5 zu sehen ist.

Am Arduino wurden für diese Signale die Pins 9, 10 und 11 verwendet, wodurch jede IMU einzeln identifiziert und angesteuert werden kann. Es werden bei diesem Design insgesamt sechs Leitungen pro IMU zum Arduino benötigt, damit sind noch vier Pole des Flachbandkabels unbesetzt. Diese lassen sich, wie hier geschehen, nutzen, um zusätzliche Leitungen für die Datenverbindungen zu ziehen. So sind beispielsweise die Taktgeber-Pins an den IMUs je über zwei Kabel mit dem Taktgeber-Pin am Arduino verbunden. Dies vermeidet Fehler durch Kabelbrüche oder -Knicke und sorgt für zusätzliche Stabilität und Zuverlässigkeit der Anbindungen.

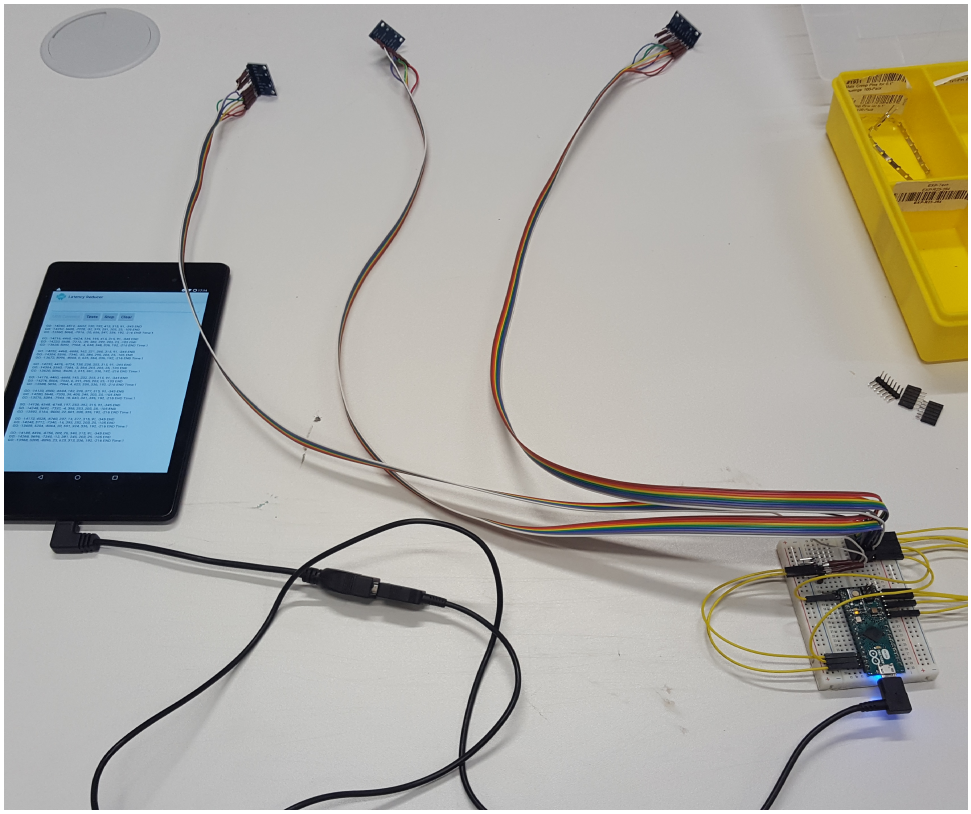


Abbildung 3.6: Zu sehen ist der finale Anschluss der IMUs am Arduino. Die gelben Kabel werden genutzt, um die Pins des Arduinos an eine günstige Position auf dem Steckbrett weiterzuleiten, an welcher sie passend angeordnet werden können. Der Arduino ist über einen USB-OTG Adapter mit einem Google Nexus 7 Tablet verbunden.

Um die Messeinheiten nun an den vorgesehenen Stellen an der Hand platzieren zu können, wird ein fingerloser Handschuh verwendet. So kann garantiert werden, dass die beiden IMUs bei jedem Anwender an der richtigen Stelle sitzen. Damit sie nicht verrutschen, werden sie mit dem Handschuh vernäht, zu sehen in Abbildung 3.7, und anschließend mit nicht-leitendem Gewebeband überklebt. Die Pins der IMUs durchstechen das Klebeband, damit sie weiterhin zugänglich sind und die Kontakte der IMUs trotzdem geschützt sind. Außerdem kann das Klebeband dazu verwendet werden, den Handschuh in seiner Enge am Finger und Handgelenk zu verstellen, falls die Finger- und Handgröße der Anwender stark variiert. Die dritte IMU wird mit Klebeband an einem gewöhnlichen Touchstylus befestigt. Die Platzierung der IMUs ist damit abgeschlossen, jedoch muss das Steckbrett mit dem Arduino noch befestigt werden. Dafür eignet sich ein Sportarmband mit Handyhalterung in der Größe des Steckbretts. Auf dieser kann das Steckbrett verklebt werden, damit es bei der Nutzung nicht im Weg ist. Sind die auf dem Handschuh und Stylus platzierten IMUs mit dem Arduino, auf dem Steckbrett, auf dem Sportarmband, verbunden, so ist der Prototyp fertiggestellt.

Abbildung 3.7: Zu sehen ist die Befestigung der IMU am Zeigefinger. Sie wurde fest mit dem Handschuh, durch Nutzung der Pinzwischenräume und Chipaussparungen, vernäht, um einen festen Sitz zu garantieren.

In Abbildung 3.8 ist der fertige Prototyp mit allen Komponenten dargestellt. Vor der Nutzung müssen Anwender lediglich das Armband am Oberarm anlegen, den Handschuh anziehen und den Arduino, über ein USB-Kabel, mit einem OTG Adapter, zu einem Android-Gerät, mit entsprechender Schnittstelle, verbinden.

3.3 Zusammenfassung

In den vorherigen Abschnitten wurden Designkonzepte beschrieben, um externe inertiale Messeinheiten mit einem Android-Gerät zu verbinden. Nach den Kriterien Hochfrequenz, Synchronizität und Fehlerlosigkeit wurden Komponenten und Technologien ausgewählt, die diese Kriterien am besten erfüllen. Daraus wurde ein finales Design erarbeitet, eine Software-Schnittstelle erzeugt und die Tragbarkeit des Prototypen hergestellt. Bei Tests mit einem Google Nexus 7 Android-Tablet stellte sich heraus, dass die durchschnittliche Frequenz, um Messdaten aller Sensoren zu erhalten, bei etwas über 600Hz liegt, wodurch durchschnittlich alle 1,66ms neue Daten von den Sensoren am Tablet ankommen. Mit diesem Prototypen sollen in der folgenden Studie Messdaten gesammelt werden. Auch später wird der Prototyp dazu genutzt werden, kontinuierlich inertiale Sensordaten, für die Vorhersage von Touchtrajektorien, auf ein Tablet zu übertragen.

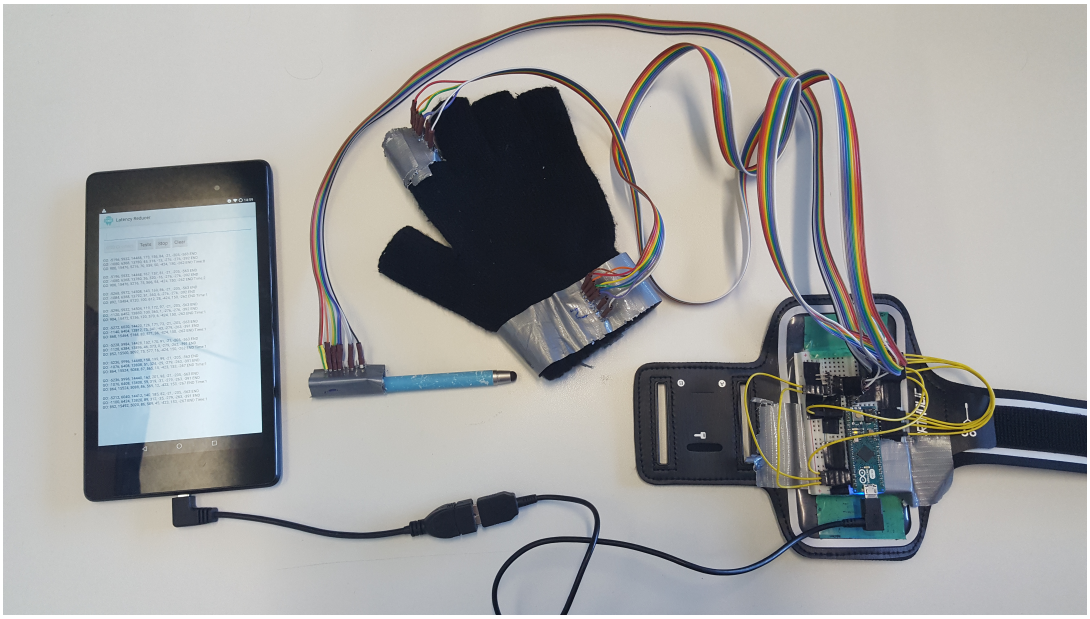


Abbildung 3.8: Die Abbildung zeigt alle Komponenten des fertigen Prototyps mit ihren festen Verbindungen. Nach dem Anlegen der einzelnen Teile und dem Verbinden des Tablets, wie auf dem Bild, können die, in Echtzeit gemessenen Sensordaten des Handschuhs, auf dem Tablet genutzt werden.

4 Studie zur Datenerhebung

Für die Erstellung von neuronalen Netzen, zur Vorhersage von Touchpositionen, müssen Daten, aus welchen das neuronale Netz lernen kann (dieser Vorgang wird Training genannt), zur Verfügung stehen. Diese Daten werden in der vorgestellten Studie erhoben. Die neuronalen Netze sollen sowohl auf den Positionen vorheriger Toucheingaben trainiert werden, als auch auf den Messwerten der drei Sensoren, der drei IMUs, des im letzten Kapitel konstruierten Prototyps. Um diese Daten zu erhalten, wurden Teilnehmer gebeten, mit dem angelegten Prototypen bestimmte Aufgaben in einer Android-Applikation auf dem Google Nexus 7 Tablet durchzuführen, während auf dem Tablet die entsprechenden Messdaten erfasst und gespeichert wurden. Die folgenden Unterkapitel beschreiben diesen Vorgang im Detail.

4.1 Erhebungsmethode

Die Erhebungsmethode beschreibt den Aufbau und Ablauf der Studie. Die grundlegende Erhebungsmethode wird unterteilt in die Beschreibung des Studiendesigns und die Beschreibung des Ablaufs der Studie, für die einzelnen Teilnehmer. Weiterhin wird die Protokollierung der erfassten Daten beschrieben, sowie die von den Teilnehmern durchgeführten Aufgaben und die Auswahl der Teilnehmer an sich.

Design

Bei der Studie handelt es sich um ein experimentelles Labordesign. Bei einem Laborexperiment werden möglichst viele unabhängige Variablen kontrolliert. Bei dieser Studie sind beispielsweise das Touchgerät, der externe Prototyp als Sensor, die zu nutzende Hand und andere Umstände fest vorgegeben. Des Weiteren wurde ein „within-subjects“-Design verwendet, was bedeutet, dass jeder Teilnehmer alle Aufgaben unter allen Bedingungen durchführen muss. Oft wird dieses Design in dieser Verwendung auch als „repeated measures“-Design bezeichnet, bedeutet aber dasselbe. Jeder Teilnehmer muss im Verlauf der Studie zweimal drei Aufgaben durchführen, einmal während der Nutzung des Touchstylus des Prototyps und einmal nur mit dem Zeigefinger. Dadurch ergibt sich ein experimentelles 3 x 2 Design. Die unabhängigen Variablen sind damit die Aufgaben und die Eingabemethode, also Finger oder Stylus. Die Reihenfolge der Aufgaben ist dabei über ein „balanced Latin square“-Design [CG80] ausgeglichen worden. Durch diesen Ausgleich bekommen immer eine konstante Anzahl Teilnehmer die gleiche

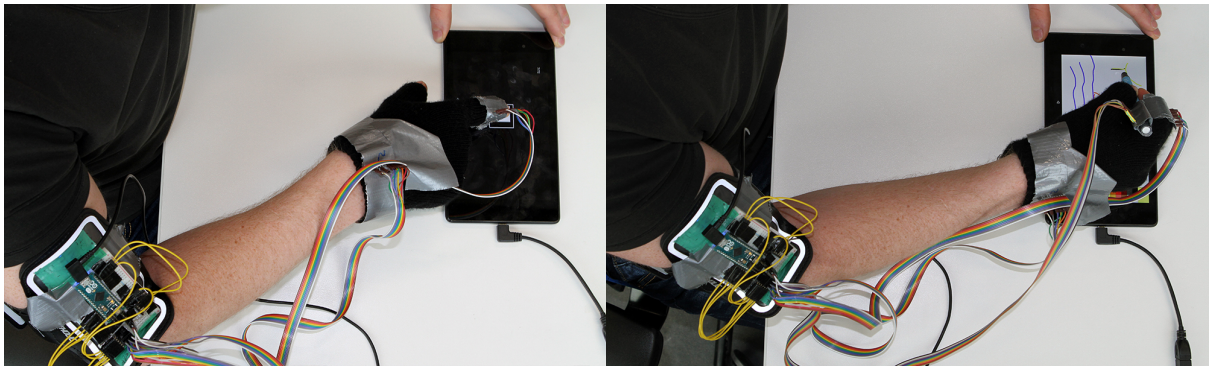


Abbildung 4.1: Ein Teilnehmer führt die Studie mit dem Prototypen durch. Auf dem linken Bild wird ein Fitts' Law-Test mit dem Finger und rechts eine Zeichenaufgabe mit dem Stylus durchgeführt.

Reihenfolge der Aufgaben. Das bedeutet, dass jede Aufgabe gleich oft als erstes, zweites, drittes, und so weiter, ausgeführt wird. Auf diese Weise werden Sequenzeffekte, beziehungsweise Lerneffekte zwischen Aufgaben irrelevant. Weiterhin wurde immer zwischen Finger- und Stylusaufgaben gewechselt, damit auch hier keine Sequenzeffekte auftreten [HSE10; Hub87; MS95; SR13; Sch11].

Prozedur

Die Studie dauerte pro Teilnehmer etwa eine Stunde und hatte folgenden Ablauf: Jeder Teilnehmer wurde nach dem Unterzeichnen einer Einverständniserklärung gebeten, den Handschuh mit den Inertialsensoren anzulegen und erhielt eine kurze Unterweisung. Dabei wurde erklärt, dass es sich um eine Studie zur Erhebung von Inertialdaten und Touchdaten handelt und wie die Aufgaben aufgebaut sind. Anschließend bekam jeder Teilnehmer eine numerische, anonymisierte ID zur späteren Zuordnung der Daten und eine feste Reihenfolge der Aufgaben nach dem „balanced Latin Square"-Design [CG80]. Nach dem Start der Aufgaben durch den Betreuer wurde die Zeit gestoppt und die Teilnehmer durften beginnen. In Abbildung 4.1 sind Bilder einer laufenden Studie abgebildet. Zwischen jeder Aufgabe wurde vom Betreuer der Studie überprüft, ob ein Fehlverhalten der externen Inertialsensoren feststellbar ist, etwa durch das Lockern von Verbindungen durch rapide Bewegungen. Dazu wurden, über einen Zeitraum von 20 Sekunden, stichprobenartig Messwerte betrachtet. Bei Fehlern wurde die USB-Verbindung zum Gerät neu aufgebaut und gegebenenfalls die letzte Aufgabe wiederholt, was später im Abschnitt über die Teilnehmer näher beschrieben wird. Während den Aufgaben wurden die Hardware-Tasten am Gerät über die Android-Applikation deaktiviert, um Fehler zu vermeiden (manche Teilnehmer drückten während den Tests die Hardwaretasten, beim Versuch, das Tablet mit der linken Hand festzuhalten) und der Flugmodus des Android-Betriebssystems wurde aktiviert, um Performanzeinbrüche zu reduzieren. Die Aufgaben wurden alle im Sitzen an einem Tisch mit angelegtem Sensorhandschuh durchgeführt.

4.1.1 Protokollierung der Daten

Während der Studie wurden Daten aufgezeichnet. Als Touchdaten wurden alle registrierten Touchereignisse des Tablets, während den Aufgaben, festgehalten, die das Android-System mit einer Frequenz von 60Hz lieferte. Dafür wurden die X- und Y-Koordinaten der Positionen auf dem Touchscreen, ein Zeitstempel mit der Systemzeit in Nanosekunden, ein Zeitstempel mit der Ereigniszeit in Millisekunden, die Größe der Fläche der Berührung und die Art des Ereignisses gespeichert. Als Ereignisarten liefert Android drei mögliche Optionen: „ACTION_DOWN“ für den Beginn einer neuen Touchtrajektorie, „ACTION_MOVE“ für das Verändern der Touchberührungsposition und „ACTION_UP “ für das Beenden einer Touchtrajektorie durch das Ende einer Touchberührung[1].

Für die IMU-Daten wurden die Messwerte gespeichert, die der IMU-Handschuh über die USB-Schnittstelle an das Android-Gerät, mit einer Frequenz von etwa 600Hz, lieferte. Für jedes erhaltene Datenpaket wurde der Zeitstempel mit der Android-Systemzeit in Nanosekunden erfasst, wie auch bei den Touchdaten. Zusätzlich wurde erfasst, wie lange die Daten im USB-Buffer des Arduino verweilt haben, um eventuell später Zeitstempelkorrekturen vorzunehmen. Von allen drei IMUs wurden die Werte aller drei Achsen aller drei Sensoren gespeichert. Demnach wurden 27 Datenwerte pro Messwertpaket exklusive der Zeitstempel gespeichert. Die Magnetometerwerte wurden bei jedem neuen Paket mitgesendet und gespeichert, auch wenn sie sich, aufgrund der geringeren Aktualisierungsfrequenz, nicht verändert hatten.

Auch die Sensor-Daten der im Google Nexus 7 verbauten Inertialsensoren wurden erfasst. Das Gerät verfügt über je ein dreiachsiges Accelerometer, Gyroskop und Magnetometer, welche über eine Android-Schnittstelle sofort bei Änderungen abgefragt werden können[2]. Bei durchgeführten Tests, vor der Studie, konnten das Accelerometer und Gyroskop mit einer Frequenz von 200Hz und das Magnetometer mit 50Hz abgefragt werden. Dies entspricht den Mindestanforderungen, für die von Google definierten Anforderungen, an Hochgenauigkeitssensoren [Rud15]. Bei jeder Sensoraktualisierung wurden die Messwerte aller drei Achsen jedes Sensors gespeichert, zusätzlich die Android-Systemzeit in Nanosekunden, die Ereigniszeit in Nanosekunden und welcher Sensor das Ereignis ausgelöst hatte. Dadurch stehen zu jedem Zeitstempel immer die aktuellsten Messwerte jedes Sensors zur Verfügung, was später beim Verbinden aller Daten Komplikationen vermeidet.

Beim Aufzeichnen der Daten werden pro Aufgabe drei Dateien auf dem Gerät erstellt. Eine Datei enthält die gespeicherten Touchdaten, eine die externen IMU-Daten und die dritte die internen Sensordaten, jeweils wie oben beschrieben. Jede Datei wird mit der Teilnehmer-ID, Aufgabenart, Zeitstempel und dem Kürzel des verwendeten Eingabegeräts (Finger oder Stylus) versehen. Dadurch sind die aufgezeichneten Daten eindeutig den Teilnehmern, Aufgaben

[1] Referenz für Touchereignisse auf Android: https://developer.android.com/reference/android/view/MotionEvent.html

[2] Android-Schnittstelle zum Abfragen der Sensorwerte: https://developer.android.com/reference/android/hardware/SensorListener.html

Abbildung 4.2: Die einzelnen Bilder stellen vier Beispiele der erstellten Zeichnungen der Teilnehmer in der Android-Applikation dar.

und Zeitpunkten zuordnungsbar. Ebenso können zwei getrennte Datensätze erstellt werden, einer für Daten, welche mit dem Stylus erzeugt wurden und der andere für Daten mit dem Finger. Das ist wichtig, da sich Toucheingaben je nach Eingabegerät unterscheiden können und die IMU-Daten der Stylus-IMU des Prototyps nicht in das spätere Training des neuronalen Netzes für Fingereingaben einfließen dürfen, da der Stylus bei Fingereingaben nicht verwendet wurde.

4.1.2 Aufgaben

Für das Aufzeichnen der Daten wurden drei verschiedene Aufgaben pro Teilnehmer durchgeführt. Diese folgen dem Vorbild der Arbeit von Henze et al. [HFS16]. Das Aufzeichnen der Mess- und Touchdaten verlief für jede Aufgabe gleich, wie gerade beschrieben. Jede Aufgabe wurde von jedem Teilnehmer zweimal durchgeführt, einmal unter Verwendung des Stylus und einmal unter Verwendung des Zeigefingers.

Zeichenaufgabe

Die ungebundenste der drei Aufgaben war eine Zeichenaufgabe. Dabei wurden die Teilnehmer gebeten, mit der jeweiligen Eingabemethoden ein Bild ihres letzten Urlaubs zu malen. Für eine

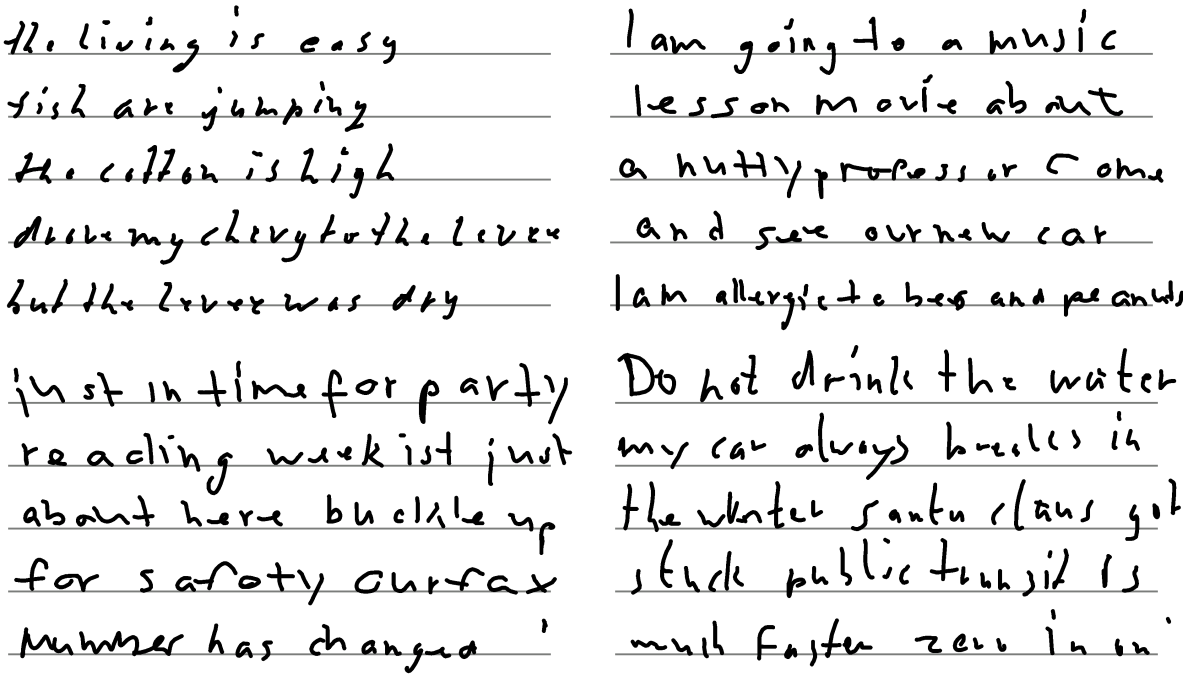


Abbildung 4.3: Die einzelnen Bilder stellen vier Beispiele der geschriebenen Phrasen der Teilnehmer in der Android-Applikation dar.

Zeichenaufgabe wurde ein Zeitfenster von 10 Minuten vorgegeben. Wenn die Teilnehmer ihr Bild fertiggestellt hatten, wurden sie gebeten, ein weiteres zu zeichnen, bis das Zeitlimit von 10 Minuten erreicht wurde. Hatten Teilnehmer keine eigene Idee, wurde ihnen geraten, ein Bild einer tropischen Insel zu malen. Das Ziel dieser Aufgabe war es, möglichst viele Daten von typischen Zeichenbewegungen zu erhalten, welche theoretisch keinen festen Mustern folgen. In der Android-Applikation wurde Nutzern zum Zeichnen der ganze Bildschirm, mit einem weißen Hintergrund, zur Verfügung gestellt, bis auf die 200 letzten Pixel jeder Pixelreihe. Dort wurde ein Menü platziert, in welchem Nutzer Farben und Strichstärken wählen konnten. Nutzer wurden animiert, möglichst den ganzen freien Bildschirmbereich auszunutzen, um Daten aus jedem Bildschirmbereich zu erhalten. Zusätzlich zu den Messdaten wurde je eine Bitmap (Rastergrafik, in welcher die Farbe jedes einzelnen Pixels eines Bildes gespeichert wird) der gezeichneten Bilder gespeichert, wie beispielsweise die vier Bilder in Abbildung 4.2.

Schreibaufgabe

Bei der Schreibaufgabe wurden die Teilnehmer dazu aufgefordert, mit der jeweiligen Eingabemethode möglichst viele vorgegebene Phrasen möglichst schnell, aber leserlich, aufzuschreiben. Dafür wurde ebenfalls ein Zeitfenster von 10 Minuten veranlasst. Die gewählten Phrasen stammen aus dem Phrasenset von MacKenzie und Soukoreff [MS03]. Daraus wurden zufällige Phrasen gewählt, bis das Zeitlimit erreicht war. Die Phrasen bestehen nur aus englischen

Sätzen ohne Satzzeichen. Die Intuition bei dieser Aufgabe war, dass Nutzerdaten typische Muster enthalten, welche beim Zeichnen von Buchstaben häufig vorkommen. Für diese Aufgabe wurde Teilnehmern eine weiße, linierte Touchoberfläche in der Android-Applikation zur Verfügung gestellt, die den ganzen Bildschirm bedeckt. Teilnehmer wurden aufgefordert, den ganzen Bereich des Bildschirms auszunutzen, um möglichst vielseitige Daten zu erhalten. Zusätzlich zu den Messdaten wurde je eine Bitmap der geschriebenen Phrasen gespeichert, von denen ein beispielhafter Auszug in Abbildung 4.3 dargestellt wurde.

Fitts' Law-Aufgabe

Die Fitts' Law-Aufgabe folgt dem Vorbild von Jota et al. [Jot+13]. Dabei handelt es sich um eine Implementierung eines Fitts' Law-Tests für Ziehaufgaben auf Touchscreens, welchen MacKenzie für Mensch-Compter Interaktionen definierte [Mac92]. Die Version von MacKenzie basiert auf den Forschungen von Fitts, welche noch nicht für Touchscreens ausgelegt war [Fit54]. Ursprünglich waren Fitts' Law-Tests nur für Zeigeaufgaben gedacht, nicht für Ziehaufgaben, deshalb die Abwandlung von Jota et al. [Jot+13]. Fitts' Law besagt, dass die Zeit, die es dauert, ein Ziel zu erreichen, von der Distanz (D) und der Breite (W) des Ziels abhängt. Nach Fitts lässt sich für jede Aufgabe über die Distanz (D) und Breite (W) des Ziels ein „index of difficulty" (ID) definieren, der angibt, wie schwierig die Aufgabe ist. Über diesen lässt sich abschätzen, wie lange Nutzer für die Bewältigung der Aufgabe brauchen werden [Fit54]. Nach der Abwandlung von MacKenzie wird dieser Index über die Gleichung 4.1 berechnet [Mac92].

$$ID = \log_2(\frac{D}{W} + 1) \tag{4.1}$$

Mit Hilfe dieser Formel lässt sich weiterhin die Performanz der Anwender in Bits pro Sekunde berechnen, wenn ihre Zeit zur Bewältgung der Aufgabe (Nach Fitts: „movement time", Abkürzung: MT) aufgezeichnet wird [Mac92]. Die Formel 4.2 berechnet den „index of performance" nach MacKenzie, welcher auch als Durchsatz gesehen werden kann [Fit54; Mac92].

$$IP = \frac{ID}{MT} \tag{4.2}$$

Während Fitts' Law-Tests, im Verlauf dieser Studie, wurden Teilnehmer gebeten, auf dem Touchscreen des Google Nexus 7, mit dem angelegten Prototypen und der jeweiligen Eingabemethode, ein kleines Quadrat in ein größeres zu ziehen, wie in Abbildung 4.4 zu sehen. Dies sollte 72 Mal hintereinander geschehen, mit verschiedenen Breiten für das Zielquadrat und verschiedenen initialen Entfernungen zwischen dem kleinen Quadrat und dem Zielquadrat. Wie bei Henze et al. wurden drei verschiedene Breiten für das Ziel (305,3 Pixel, 407,1 Pixel, 508,8 Pixel enspricht folgenden Breiten beim Google Nexus 7: 24mm, 32, 40mm) und drei verschiedene Entfernungen zum Ziel (356,2 Pixel, 864,9 Pixel, 1526,4 Pixel enspricht folgenden Distanzen beim Google Nexus 7: 28mm, 68mm, 120mm) gewählt [HFS16]. In Abbildung 4.5

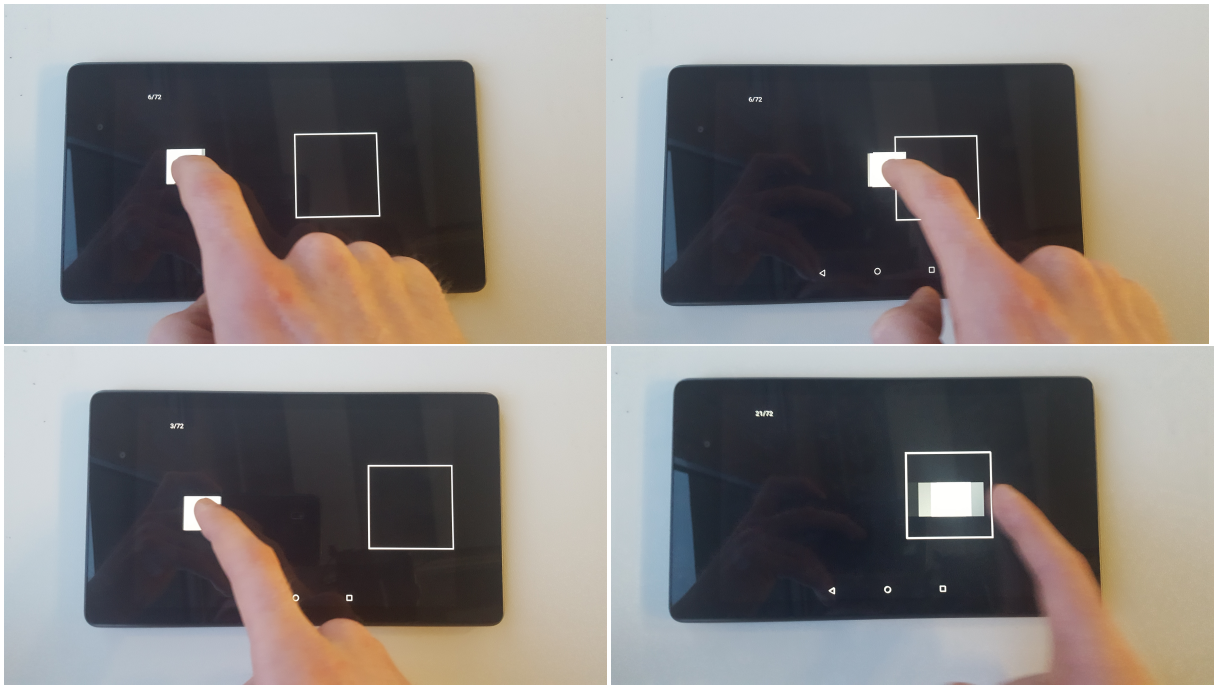


Abbildung 4.4: Ein Teilnehmer führt Fitts' Law-Aufgaben in der Android-Applikation aus. Für die Bilder wurde der Handschuh mit den Sensoren weggelassen.

sind alle Größen, vergleichend für eine Zielposition, auf dem Nexus 7 dargestellt. Daraus resultieren neun verschiedene Kombinationen von Aufgabenaufbauten. Diese neun Kombinationen kommen in randomisierter Reihenfolge nacheinander vor. Anschließend wird der Vorgang sieben Mal wiederholt. Die Positionen des Ziels und des kleinen Quadrates werden bei jeder Teilaufgaben variiert, ihre Mittelpunkte befinden sich jedoch immer auf einer gemeinsamen, horizontalen Achse des Bildschirms, mit einem der definierten Abstände. Insgesamt muss jeder Teilnehmer demnach $8 \times 9 = 72$ Einzelaufgaben erledigen, um einen Fitts' Law-Test abzuschließen.

Eine Aufgabe beginnt mit dem Berühren des kleinen Quadrats und endet mit dem Loslassen. Dazwischen wird die Zeit gestoppt. Befindet sich das kleine Quadrat beim Loslassen nicht im großen Quadrat, oder berührt noch dessen Rand, wird ein Fehler registriert. Den Nutzern wird nach jeder Einzelaufgabe angezeigt, wie viel Prozent der bisherigen Aufgaben fehlerhaft waren. Sie wurden vor dem Ausführen der Tests aufgefordert, die Aufgaben so schnell und fehlerlos wie möglich zu lösen. Lag die Fehlerrate der Teilnehmer unter 5%, wurden sie zusätzlich motiviert, die Aufgaben schneller zu lösen. Für den Fitts' Law-Test wurde kein Zeitlimit festgelegt. Es wurde eine eigene Touchoberfläche in der Applikation angelegt, welche den ganzen Bildschirm nutzt und nur die beiden Quadrate zeichnet (vergleiche Abbildung 4.5). Ziel dieser Aufgaben war bei dieser Studie wieder das Sammeln von Daten. Die Daten sollten diesmal gerade gerichtete, schnelle Bewegungen charakterisieren, da solche Bewegungen bei den anderen beiden Aufgaben weniger häufig vorkommen.

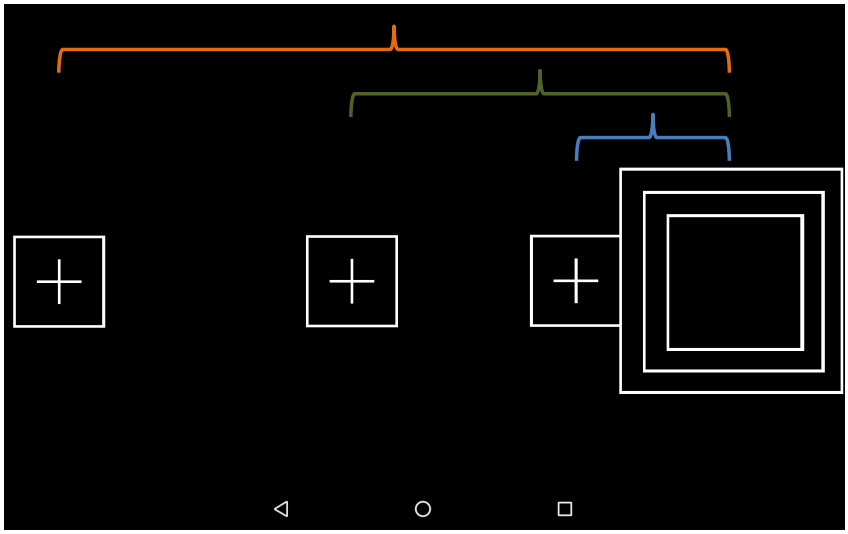


Abbildung 4.5: Alle drei Breiten und Distanzen wurden, für eine Zielposition, vergleichend dargestellt. Die Klammern veranschaulichen die drei Distanzen (blau = 28mm, grün = 68mm, orange = 120mm). Alle Mittelpunkte teilen sich eine gemeinsame, horizontale Achse.

Bei der hier erstellten Implementierung werden zusätzliche Daten bei den Fitts' Law-Tests erhoben. Eine Datei wird erstellt, welche 72 Zeilen enthält. Darin befinden sich für jede Teilaufgabe mehrere Werte, um die Aufgaben eindeutig differenzieren zu können: Ein Zeitstempel mit der Systemzeit des Android-Geräts in Nanosekunden, die Breite des Ziels in Pixeln, die Distanz zum Ziel in Pixeln, die X- und Y-Koordinaten der Position des Zielquadrats, die benötigte Zeit zur Lösung der Aufgabe in Millisekunden und ein Vermerk, ob das Ziel getroffen wurde, oder nicht. Diese Daten fallen bei dieser Studie nicht ins Gewicht, sondern wurden erst bei der späteren Evaluation verwendet.

4.1.3 Teilnehmer

An der Studie nahmen 18 Teilnehmer teil. Neun davon waren weiblich und neun männlich. Zwölf wurden zufällig am Campus der Universität Stuttgart ausgewählt, die restlichen sechs nach Terminvereinbarung in 71032 Böblingen. Die ersteren elf Teilnehmer waren alle Studenten verschiedener technischer Studiengänge in verschiedenen Semestern. Die restlichen sieben Teilnehmer waren in einem festen Arbeitsverhältnis eines nicht technischen Bereichs. Das Alter der Teilnehmer reichte von 19 Jahren bis 61 Jahren mit einem Durchschnitt von 26,8 Jahren (SD = 11,6 Jahre). Alle Teilnehmer waren Rechtshänder und gaben an, keine motorischen Störungen zu besitzen. Die Handgrößen, gemessen vom Handgelenk zur Mittelfingerspitze,

reichten von 15,8cm bis 20,0cm bei einem Durchschnitt von 17,5cm (SD = 1,2cm). Damit waren die Handgrößen der Teilnehmer breit gefächert, mit einzelnen Größen weit unter und über dem allgemeinen Durchschnitt (18,9cm bei Männern und 17,2cm bei Frauen)[3] [Pos00]. Weiterhin wurde von allen Teilnehmern die Breite und Länge des verwendeten Zeigefingers protokolliert, was eine durchschnittliche Breite von 1,9cm und eine Länge von 7,8cm offenbarte.

Alle Teilnehmer hatten im Zeitraum der Studie täglich einen Touchscreen verwendet. Nur sechs Teilnehmer verwendeten regelmäßig einen Touchstylus. Von den sechs verwendete nur eine Person täglich einen Stylus, eine Person wöchentlich und die restlichen vier monatlich. Die übrigen Teilnehmer gaben an, nie einen Stylus für Touchgeräte zu verwenden. Alle Teilnehmer nutzten ihre eigenen Touchgeräte regelmäßig für das Surfen im Internet und zum Chatten. Dreizehn Teilnehmer schrieben weiterhin regelmäßig E-Mails, neun spielten, vier editierten Texte und je eine Person programmierte und bearbeitete Bilder auf ihrem Touchgerät.

Vor der Studie wurde den Teilnehmern erklärt, dass sie jederzeit Pausen einlegen können. Kein Teilnehmer legte eine Pause während der Studie ein. Weiterhin war es den Teilnehmern gestattet, das Tablet während den Aufgaben in die Hand zu nehmen, falls sie dies als bequemer empfanden. Nur ein Teilnehmer nutze diese Möglichkeit, bei den restlichen Teilnehmern ruhte das Tablet auf einem Tisch. Am Ende wurden alle Teilnehmer mit 10€ für ihre Teilnahme entlohnt. Bei einem Teilnehmer kam es während einer Aufgabe zum Ausfall eines Accelerometers, durch das Berühren eines Kabels mit einer nicht isolierten Stelle an einer IMU. Der Teilnehmer wiederholte die entsprechende Aufgabe freiwillig. Bei vier weiteren Teilnehmern kam es, bei einzelnen Aufgaben, zum Ausfall eines Magnetometers, aufgrund von Kabelannährungen. Die Daten wurden gekennzeichnet, die Verbindung zum Magnetometer neu hergestellt und mit der nächsten Aufgabe fortgefahren. Das Problem konnte durch eine individuelle Fixierung der Kabel für jeden Teilnehmer später umgangen werden.

4.2 Ergebnisse

Im Verlauf der Studie wurden 503647 Touchberührungen durch Fingeraufgaben und 500020 Touchberührungen durch Stylusaufgaben registriert und protokolliert, mit allen Daten, wie oben erläutert. Die erwartete Frequenz aufeinander folgender Touchberührungen beträgt 60Hz für das Nexus 7, was bedeutet, dass die zeitliche Differenz zwischen zwei registrierten Touchberührungen einer Bewegung im Durchschnitt 16,66ms betragen sollte.

In den aufgezeichneten Daten für Fingeraufgaben betrug die zeitliche Lücke zwischen den Zeitstempeln im Durchschnitt 17,28ms (SD = 12,99ms, Median = 16,82ms). 55,49% der Daten hatten eine zeitliche Differenz von 16,66ms oder weniger, wie erwartet für eine Frequenz von 60Hz. 42,27% hatten eine zeitliche Differenz zwischen 16,67ms und 30ms. Die restlichen 2,23% wiesen eine zeitliche Lücke von mehr als 30ms auf, hatten demnach eine Abtastrate

[3] Durchschnitt der Handgröße: http://www.theaveragebody.com/average_hand_size.php

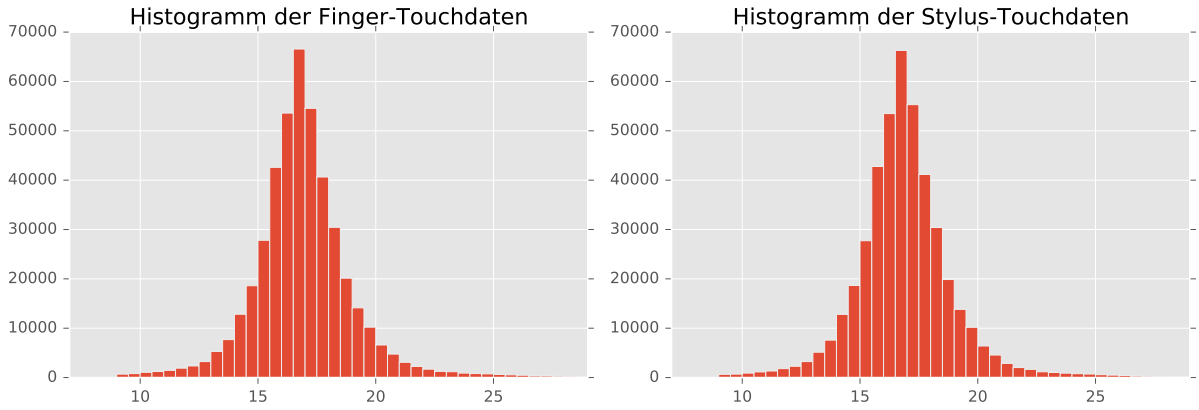


Abbildung 4.6: Die Abbildungen zeigen die Histogramme der zeitlichen Differenzen der gesammelten Touchdaten. Links ist das Histogramm der Daten, die mit dem Finger gesammelt wurden und rechts das der Stylus-Daten. Die x-Achse gibt die zeitliche Differenz der Daten in Millisekunden an, die y-Achse die Anzahl der gesammelten Berührungen mit der jeweiligen Differenz der x-Achse.

von weniger als 30Hz. Diese Daten wurden aufgrund der schlechten Abtastrate aus dem Datensatz herausgefiltert. Bei den Stylus-Touchdaten betrug die zeitliche Lücke zwischen den Zeitstempeln im Durchschnitt 17,25ms (SD = 7,42ms, Median = 16,82ms). 55,32% der Daten hatten eine zeitliche Differenz von 16,66ms oder weniger, 43,34% hatten eine zeitliche Differenz zwischen 16,67ms und 30ms und die restlichen 1,34% wiesen eine zeitliche Lücke von mehr als 30ms auf, weshalb auch diese aus dem Datensatz herausgefiltert wurden. In Abbildung 4.6 sind die Histogramme der Verteilungen, der zeitlichen Differenzen der Daten, abgebildet. Dabei lassen sich Normalverteilungen der gesammelten Daten erkennen, was für ein normales, additives Rauschen der Daten, ohne zusätzliche Störungen, spricht.

Die Daten der IMUs sind weitaus zahlreicher, wie zu erwarten bei der hohen Abtastfrequenz, die im letzten Kapitel erreicht wurde. Während der Benutzung des Fingers wurden 14216904 IMU-Pakete gesammelt, mit den Daten aller drei externen IMUs. Die Differenz ihrer Zeitstempel beträgt im Schnitt 1,71ms (SD = 5,67ms, Median = 1,31ms). Die Durchschnittsdifferenz entspricht knapp den 600Hz (hier durchschnittlich 585,14Hz), welche in Kapitel 3 erreicht wurden, die Standardabweichung (SD = 5,67ms) ist allerdings signifikant höher. Von den gesammelten Daten wurden 24,37% mit einer zeitlichen Differenz von weniger als 1ms aufgezeichnet, 74,27% mit einer Differenz zwischen 1ms und 5ms, 1,25% mit einer Differenz von 5ms bis 10ms und 0,11% mit einer höheren Differenz. Herausgefiltert wurden keine dieser Daten. Die Magnetometerfrequenz betrug im Schnitt 80,13Hz.

Mit dem Stylus ergeben sich analog folgende Zahlen: Es wurden 14119369 Datenpakete während der Benutzung des Stylus erhoben. Die Zeitstempeldifferenzen betragen durchschnittlich 1,73ms (SD = 7,06ms, Median = 1,31ms). Wieder ist die Standardabweichung (SD = 7,06ms) signifikant höher als in Kapitel 3. Von den gesammelten Daten wurden 24,03% mit einer zeitlichen Differenz von weniger als 1ms aufgezeichnet, 74,59% mit einer Differenz zwischen

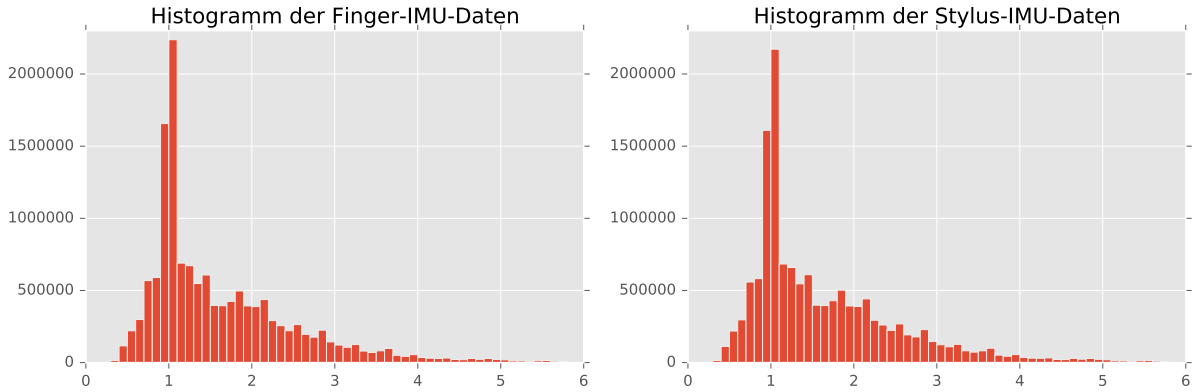


Abbildung 4.7: Die Abbildungen zeigen die Histogramme der zeitlichen Differenzen der gesammelten IMU-Daten. Links ist das Histogramm der Daten, die mit dem Finger gesammelt wurden und rechts das der Stylus-Daten. Die x-Achse gibt die zeitliche Differenz der Daten in Millisekunden an, die y-Achse die Anzahl der gesammelten Daten mit der jeweiligen Differenz der x-Achse.

1ms und 5ms, 1,28% mit einer Differenz von 5ms bis 10ms und 0,11% mit einer höheren Differenz. Herausgefiltert wurden wieder keine dieser Daten. Die Magnetometerfrequenz betrug im Schnitt 83,37Hz. Die Histogramme der IMU-Daten, für beide Datensätze, sind in Abbildung 4.7 dargestellt. Es lassen sich logarithmische Normalverteilungen der gesammelten Daten erkennen, was für ein normales, multiplikatives Rauschen der Daten, ohne zusätzliche Störungen, spricht.

Zuletzt wurden noch die Sensordaten, der internen Sensoren, des Google Nexus 7, gesammelt. Diese wurden einzeln für jeden Sensor gesammelt, da sie so vom Android-System geliefert wurden. Für das Accelerometer wurden 4736690 dreiachsige Messdaten bei Fingernutzung und 4777922 bei Stylusnutzung gesammelt. Für das Gyroskop entsprechend 4712826 und 4764985 und für das Magnetometer 1187460 und 1197885. Die Accelerometerdaten lagen, für Fingeraufgaben, zeitlich im Schnitt 5,13ms auseinander (SD = 63,00ms, Median = 4,88ms), für Stylusaufgaben entsprechend 5,10ms (SD = 25,58ms, Median = 4,88ms). Für das Gyroskop ergaben sich, für Fingeraufgaben, durchschnittlich zeitliche Differenzen von 5,13ms (SD = 63,18ms, Median = 4,85ms) und für Stylusaufgaben 5,10ms (SD = 25,67ms, Median = 4,85ms). Für das Magnetometer analog 20,47ms (SD = 125,27ms, Median = 19,99ms) und 20,36ms (SD = 50,37ms, Median = 19,98ms). Die Frequenzen der Aktualisierungen entsprechen im Durchschnitt den erwarteten Frequenzen des Tablets mit 200Hz (Accelerometer), 200Hz (Gyroskop) und 50Hz (Magnetometer), für die drei Sensoren. In Abbildung 4.8 sind je die Histogramme der einzelnen Sensoren, für beide Datensätze, dargestellt. Die gesammelten Daten aller Sensoren weisen Normalverteilungen auf, was für ein normales, additives Rauschen der Daten, ohne zusätzliche Störungen, spricht, lässt man bei der Betrachtung je die Balken ganz links außen vor. Auf diese wird in der Diskussion eingegangen.

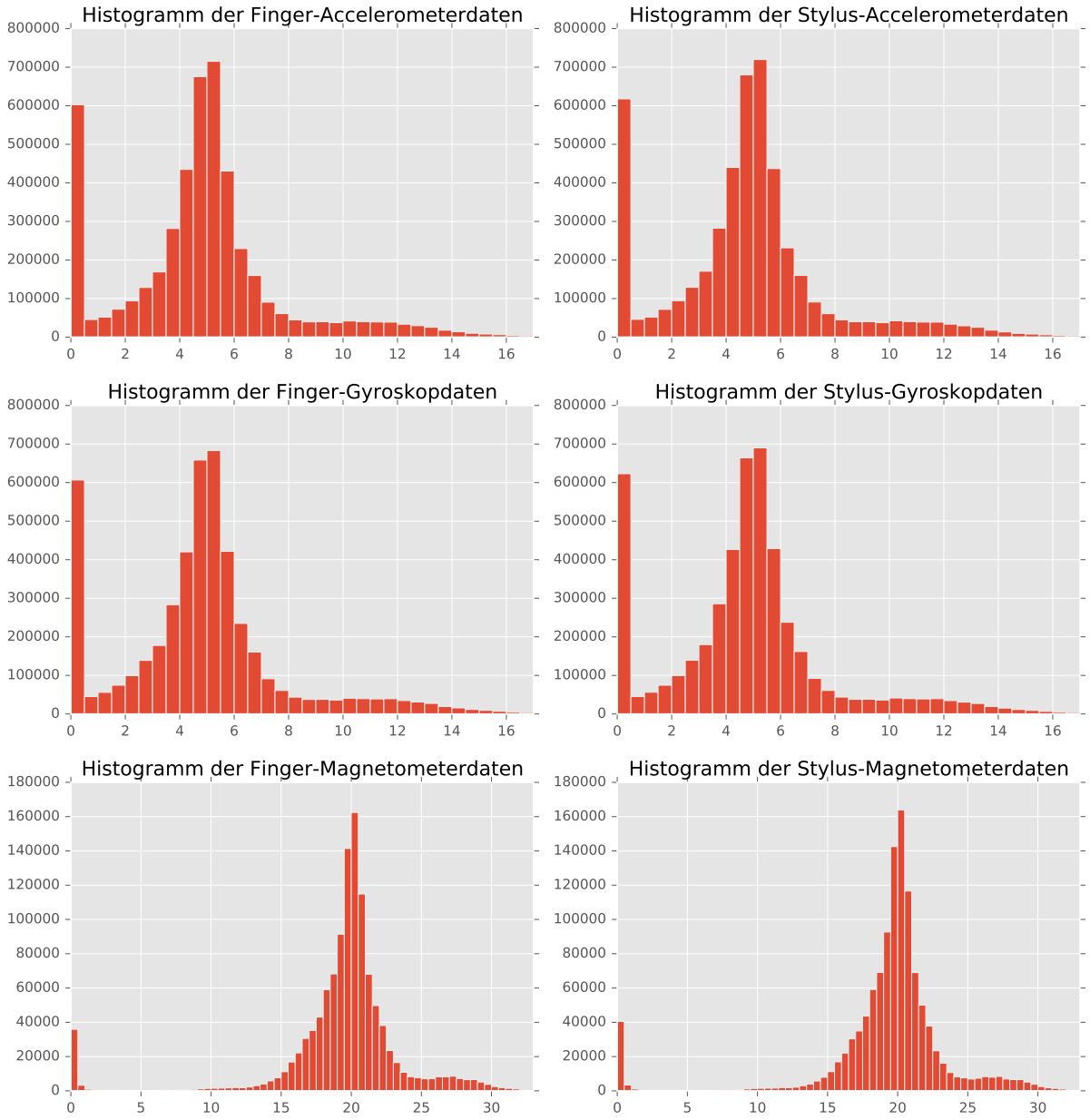


Abbildung 4.8: Zu sehen sind die Histogramme, der zeitlichen Differenzen, zwischen Messungen der Daten der internen Sensoren, des Google Nexus Tablets. Oben sind die Histogramme der Accelerometerdaten, in der Mitte die der Gyroskopdaten und unten die der Magnetometerdaten. Die Achsen entsprechen den Achsen der vorherigen Histogramme.

4.3 Diskussion

Es wurde insgesamt eine große Menge an Daten aller nötigen Messgrößen gesammelt. Die Abtastraten der Messwerte waren durchschnittlich gut. Die Histogramme zeigten Normalverteilungen für alle aufgenommenen Daten, was für geringes, störendes Rauschen in den Daten spricht. Sie eignen sich folglich für die Weiterverarbeitung im nächsten Kapitel. Trotzdem sind viele Standardabweichungen sehr hoch, bedingt durch Ausreißer. Die Touchdaten bilden später die Grundbausteine der neuronalen Netze, weshalb hier die Ausreißer aussortiert wurden.

Die IMU-Daten kamen sehr zuverlässig und hochfrequent an. In jedem Datensatz wurden nur etwas über 0,1% der Daten mit einer Frequenz von unter 100Hz abgetastet, welche als Ausreißer gewertet wurden. Nach dieser Wertung funktioniert der Prototyp zu über 99,8% zuverlässig. Auch die durchschnittliche Frequenz entspricht der aus Kapitel 3 und die Standardabweichungen sind zwar höher, liegen aber in einem akzeptablen Rahmen, wie man auch den Verteilungen in den Histogrammen entnehmen kann. Ausreißer gab es vor allen bei den Magnetometerdaten. Berührten sich Flachbandkabel und IMUs, so kam es bei vier Teilnehmern zu Fehlverhalten der Magnetometer. Diese änderten ihre Werte ab diesem Zeitpunkt nicht mehr, bis sie neu verbunden wurden. Andere Fehler, wie etwa kurzzeitige Fehlverhalten der Sensoren, wurden nicht erkannt und verbleiben, falls existent, in den Datensätzen. Bei einigen Teilnehmern berührten die Flachbandkabel einige Male den Bildschirm und es wurde fälschlicherweise eine Touchberührung erkannt und vom Tablet protokolliert. Diese wurden aufgrund der Komplexität, beim manuellen Erkennen, auch in den Datensätzen belassen, da die neuronalen Netze diese beim Trainieren, mit der größeren Menge an korrekten Daten, überdecken.

Auch bei den internen Sensordaten existieren Ausreißer. Diese gehen aber in eine andere Richtung, wie den Histogrammen zu entnehmen ist. Das Android-System erkannte manchmal eine Sensoränderung zweimal und registrierte deshalb zwei Sensoränderungsereignisse wenige Mikrosekunden hintereinander. Etwa bei 10% der hier aufgenommenen Daten lassen sich dadurch Abtastfrequenzen von über 1kHz nachweisen, welche bei Sensoren mit 200Hz beziehungsweise 50Hz nicht vorkommen sollten. Ein weiterer Kritikpunkt ist, dass diese Sensoren nur bei einer Sensoränderung registriert werden, nicht kontinuierlich, wie die IMU-Daten. Dadurch lässt sich auch die hohe Standardabweichung bei diesen Daten erklären, denn wird eine Zeit lang keine Änderung wahrgenommen, sinkt automatisch die Abtastfrequenz der Daten über die Zeit. Trotzdem stehen auch diese Daten sehr hochfrequent für das spätere Training zur Verfügung.

Weitere Fehlverhalten sind auf das Android-System zurückzuführen. Dieses ist verantwortlich für hohe Ausreißer der temporalen Differenzen der gespeicherten Daten. Ist die Auslastung des Tablets zu hoch, der Speicher während dem Anlegen einer Messdatendatei ausgelastet oder die Befehlswarteschleife voll, welche alle Android-Befehle an das System enthält und abarbeitet, so werden IMU-Daten nicht zeitnah angenommen, Touchdaten verspätet verarbeitet und interne Sensordaten nicht angefragt. Diese Probleme könnten bei zukünftigen Arbeiten durch den

Umstieg auf ein leistungsstärkeres Gerät behoben werden, wobei die Auswirkungen sich auch bei dem Nexus 7 im Rahmen halten.

Die Daten der Stylus-Aufgaben wurden zu zwei Dritteln von Personen erzeugt, welche nicht regelmäßig Styli benutzen. Auch die regelmäßigen Nutzer klagten über die Ungenauigkeit des zur Verfügung gestellten Stylus, welcher eine sehr weiche und relativ große Spitze aufweist. Demnach könnten die aufgezeichneten Daten nicht repräsentativ für andere Nutzer sein, welche regelmäßig kommerzielle Styli für ihre Touchgeräte verwenden. Der genutzte Stylus ist ein weit verbreiteter Stylus, welcher bei fast jedem Touchgerät funktioniert. Präzisere Styli sind hingegen oft auf bestimmte Touchgeräte zugeschnitten.

Die gravierendsten der eben angesprochenen Probleme betreffen lediglich die Daten der Magnetometer und internen Sensoren, welche im Training der neuronalen Netze eher eine ergänzende, beziehungsweise korrigierende Funktion erhalten sollen. Die Daten der Accelerometer und Gyroskope der IMUs konnten dafür sehr hochfrequent und zuverlässig aufgezeichnet werden, ebenso die Touchdaten des Nexus 7, aus welchen zusätzlich die Ausreißer herausgefiltert wurden. Damit sind die nötigsten Daten für das geplante Training der neuronalen Netze, im nächsten Kapitel, in ausreichender Quantität und Qualität verfügbar. Die Daten stammen von sehr unterschiedlichen Personen mit teilweise großen Altersunterschieden und einer großen Varianz bei den Handgrößen. Die Modelle, welche aus den Daten entstehen, funktionieren demnach sehr allgemein für verschiedene Personengruppen und nicht nur für bestimmte Personen mit fest definierten Handgrößen. Das Ergebnis der Studie ist eine umfangreiche Datensammlung, aufgeteilt in zwei Datensätze (Finger und Stylus). Sie bestehen aus Touch-, IMU- und internen Sensordaten, welche aber noch für sich stehen und nach Bedarf zusammengeführt werden müssen.

5 Neuronale Netze

Mit den Daten aus der vorangegangenen Studie sollen neuronale Netze erstellt werden. Dies sind Modelle aus Daten, welche genutzt werden können, um zum Beispiel Bilder zu kategorisieren oder Werte vorherzusagen. Der Modellerstellung geht ein Training auf vordefinierten Daten voraus. In dieser Arbeit werden die neuronalen Netze genutzt, um Finger- oder Stylusbewegungen, mit dem bisherigen Aufbau, auf dem Google Nexus 7 Touchtablet vorherzusagen. Dafür wird mit Hilfe des neuronalen Netzes eine lineare Regression durchgeführt [Zie03].

Zur Erstellung der hier beschriebenen neuronalen Netze wurde TensorFlow[1] genutzt, eine open-source Bibliothek für maschinelles Lernen. Die nötigen Daten, für das Training der neuronalen Netze, stammen ausschließlich aus der vorangegangenen Studie. In den folgenden Abschnitten wird die Konstruktion der hier erzeugten neuronalen Netze im Detail beschrieben. Zuerst wird dazu auf die Aufbereitung der gesammelten Daten eingegangen. Anschließend wird die Erstellung der Merkmalsvektoren erklärt, die zum Training des Netzes genutzt werden und definieren, auf Basis welcher Merkmale ein Modell gelernt werden soll. Danach wird auf die wesentlichen Aspekte des Trainingsprozesses und der dafür nötigen Strukturen eingegangen. Darauf folgend werden verschiedene neuronalen Netze verglichen und diskutiert, welche als Ergebnis dieses Vorgehens, im Rahmen dieser Ausarbeitung, entstanden. Davon ausgehend wird beschrieben, welche Modelle aufgrund welcher Kriterien für die anknüpfende Evaluation herangezogen wurden und abschließend, wie die neuronalen Netze in die Android-Applikation und das Testsystem integriert wurden.

5.1 Aufbereitung des Datensatzes

Vor der Erstellung der Merkmalsvektoren mussten zuerst die gesammelten IMU-Daten den Touchdaten aus der vorherigen Studie zugeordnet werden. Dafür wurde ein „Nearest-Neighbor"-Ansatz (deutsch: Nächste-Nachbarn-Ansatz) gewählt, welcher jedem Touchpunkt das naheste-henste IMU-Paket zuordnet, in Bezug auf die Zeitstempel der Android-Systemzeit. Dabei gab es zwei weitere Kriterien. Erstens musste der Zeitstempel der IMU-Pakete vor den Zeitstempeln der Touchdaten liegen, damit diese zusammengeführt werden können. Das ist deshalb nötig, weil im späteren Android-Programm direkt bei einer neuen Touchberührung eine Vorhersage

[1] TensorFlow-Website: https://www.tensorflow.org/

ausgelöst werden soll. Demnach werden spätere IMU-Daten bei einer Vorhersage in Echtzeit nicht beachtet und sollten auch keinen Einfluss beim Training besitzen.

Zweitens durfte die Differenz zwischen den Touch- und IMU-Zeitstempeln höchstens 10ms betragen. So wird verhindert, dass veraltete IMU-Daten für das Training genutzt werden, falls aufgrund von Fehlern keine aktuelleren IMU-Daten vorhanden sind. Touchdaten, welche kein IMU-Paket zugewiesen bekamen, das den Kriterien entsprach, wurden aussortiert. Das ergab 11856 herausgefilterte Datenpunkte für den Finger-Touchdatensatz und 7396 für den Stylus-Touchdatensatz. Die resultierenden Touchpunkte waren alle mit dem nächsten IMU-Datenpaket ihres Datensatzes (Stylus oder Finger) verbunden. Ein IMU-Datenpaket alleine hat aber keinen großen Informationsgehalt, da sich diese Daten sehr schnell verändern. Demnach wurden hier jedem IMU-Touchpunkt-Paar neun weitere IMU-Pakete zugeteilt. Ausgehend von dem zugeteilten ersten IMU-Paket wurden die neun vorherigen Pakete dazugenommen. Damit ergeben sich IMU-Touchpunkt-Paare mit je einem Touchpunkt und den naheliegensten zehn vorherigen IMU-Paketen, in jedem der beiden Datensätze (Finger-Daten und Stylus-Daten). Die Anzahl zehn stellt dabei sicher, dass immer eine konstante Anzahl dieser Daten verwendet werden, da die Eingabevektorgröße fest definiert werden muss. Außerdem wird dadurch sichergestellt, dass sich die Magnetometerdaten innerhalb von zehn Paketen, also einem Zeitraum von durchschnittlich etwas über 16ms (ausgehend von der durchschnittlichen Frequenz der IMU-Pakete), mindestens einmal geändert haben. Die Zuordnung der internen Sensordaten des Tablets erfolgte dabei analog zu der Zuordnung der IMU-Daten.

5.2 Konstruktion eines Merkmalsvektors

Wie bei Deber et al. wird die Vorhersagezeit durch die Anzahl vorhergesagter Touchereignisse definiert [Deb+15]. Bei einem 60Hz Display, zum Beispiel dem des Google Nexus 7, bedeutet die Vorhersage eines Touchpunktes demnach eine Vorhersage von 16,6ms. Zwei Punkte entsprechen analog einer Vorhersage von 33,3ms, drei Punkte 49,9ms, vier Punkte 66,6ms und n Punkte einer Vorhersage von $n * 16,6ms$. Henze et al. konnten mit der Methode von Bérard und Blanch und einer 240Hz Kamera beim Google Nexus 7 eine Latenz von 100ms nachweisen [BB13; HFS16]. Um diese Latenz fast vollständig zu kompensieren ist eine Vorhersage von sechs Touchpunkten, demnach 99,9ms, notwendig. Jedes hier erstellte künstliche neuronale Netz soll demnach sechs Touchpunkte vorhersagen. Damit neuronale Netze solch ein Verhalten im Training lernen, sind zwei Datensätze nötig, die Merkmalsvektoren, bestehend aus Eingabedaten und die Ergebnisvektoren, bestehend aus „Ground-Truth“-Daten, also Zielwerten. Die Eingabedaten spiegeln die Werte wieder, auf deren Grundlage das Netz Vorhersagen treffen soll, beispielsweise aktuelle Touchpunkte und IMU-Daten. Die Ground-Truth-Daten sind beim Training die Solldaten, welche die gewünschten Ergebnisse zu den trainierten Eingabedaten enthalten, in diesem Fall die sechs Koordinaten der Touchpunkte, welche vorhergesagt werden sollen. Das neuronale Netz lernt später beim Training ein Modell, basierend auf dem Merkmalsvektor und optimiert für die Ground-Truth-Daten.

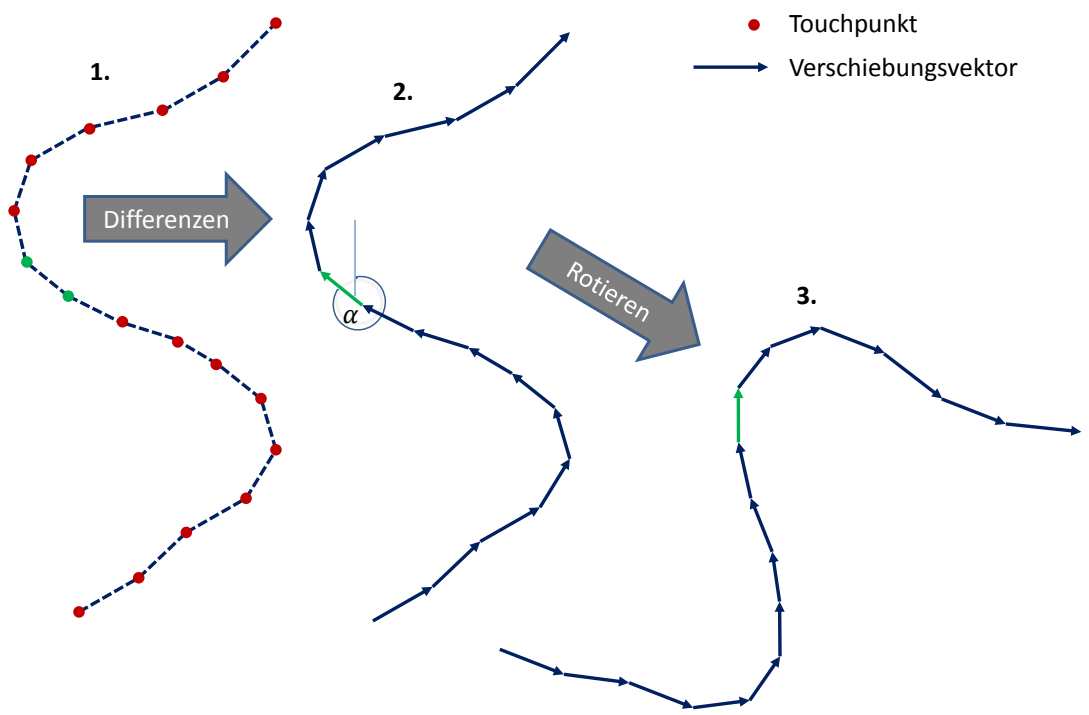


Abbildung 5.1: Die Abbildung zeigt das Prinzip, nach welchem die Touchpunkte positions- und rotationsinvariant gemacht werden. Von den Touchpunkten (1.) werden zunächst Differenzen gebildet. Von den entstehenden Vektoren, der Verschiebungen zwischen den Punkten (2.), wird der Winkel α, zwischen Vektor 10 (grün eingefärbt; Verschiebung zwischen Punkt 10 und 11, ebenfalls grün eingefärbt) und der y-Achse, berechnet. Die Vektoren werden anschließend (3.) um α gegen den Uhrzeigersinn, um den Ursprung, rotiert.

Der Merkmalsvektor, der die Eingabedaten enthält, und der zugehörige Ground-Truth-Vektor, folgten bei ihrer Konstruktion dem Vorgehen von Henze et al. [HFS16]. Die Datensätze aus dem vorherigen Kapitel wurden zuerst in Trajektorien von 17 zusammenhängenden Touchpunkten unterteilt. Dabei wurde darauf geachtet, dass die einzelnen Touchpunktzeitstempel höchstens 30ms auseinander liegen. Touchpunkte, welchem dem Ereignis „TOUCH_UP“ entsprachen, also dem Beenden einer Touchberührung, wurden nicht betrachtet und „TOUCH_DOWN“ Ereignisse durften nur am Anfang einer Trajektorie stehen. Aus den erhobenen Daten für Fingeraufgaben konnten so 131027 Trajektorien und aus den Stylusdaten 140674 Trajektorien extrahiert werden. Anschließend wurden die Trajektorien positionsinvariant gemacht, indem zwischen alle aufeinander folgenden Punkten die Differenzen berechnet wurden. Als Ergebnis blieben 16 Verschiebungsvektoren, für die Verschiebungen zwischen den 17 Touchpunkten im zweidimensionalen Raum. Diese Verschiebungen wurden anschließend rotationsinvariant

gemacht. Dafür wurde der Winkel zwischen dem zehnten Verschiebungsvektor, also die Verschiebung zwischen den ursprünglichen Touchpunkten 10 und 11, und der y-Achse berechnet. Dann wurde die komplette Trajektorie um den Ursprung (Koordinaten x = 0, y = 0) gegen den Uhrzeigersinn rotiert, sodass der zehnte Verschiebungsvektor anschließend parallel zur y-Achse verläuft. In Abbildung 5.1 wurden die einzelnen Schritte visualisiert.

Von diesen Trajektorien wurden anschließend je die letzten 6 rotierten Verschiebungsvektoren als Ground-Truth-Daten genommen und die ersten 10 als Eingabedaten. Der Trainingsvorgang erwartet zwei eindimensionale Arrays, weshalb die x- und y-Koordinaten der rotierten Vektoren jeweils hintereinander im Array platziert wurden. Der entstandene Merkmalsvektor, mit den Eingabedaten für das Training, bestand demnach aus 20 Werten mit x- und y-Koordinaten der Vektoren im Gleitkommaformat. Der Vektor mit den Ground-Truth-Daten bestand entsprechend aus 12 Werten. Je nach gewünschtem Modell mussten anschließend die benötigten IMU-Daten hinzugezogen werden. Vor dem Einbeziehen dieser Daten wurden sie skaliert, da sie aus Effizienzgründen bisher in Rohform (als zwei Byte große Short-Werte) vorlagen. Für die Accelerometer wurde die Skalierung in Formel 5.1 verwendet mit SkAcc für die fertig skalierten und RoAcc für die rohen Accelerometerwerte.

$$SkAcc = RoAcc * \frac{2.0 * 9.807 \frac{m}{s^2}}{32767.5} \tag{5.1}$$

Die Accelerometer wurden vor der erste Studie für eine Sensitivität von 2G (G = Erdbeschleunigung = 9.807 m/s^2) konfiguriert, mit welcher die Werte anschließend aufgezeichnet wurden. Um entsprechend die richtigen Werte zu rekonstruieren, mussten die gelieferten Rohwerte, durch eine Multiplikation mit 2G und eine Division durch den aufgespannten Wertebereich von Short-Werten, auf den passenden 2G-Beschleunigungswertebereich skaliert werden. Für die Gyroskope wurde entsprechend die Formel 5.2 verwendet, mit RoGyr als Platzhalter für die rohen und SkGyr als Platzhalter für die fertig skalierten Gyroskopwerte.

$$SkGyr = RoGyr * \frac{\frac{\pi}{180°} rad * 250 \frac{°}{s}}{32767.5} \tag{5.2}$$

Die Gyroskope wurden für eine Sensitivität von $250°/s$ konfiguriert. Die eingestellten Sensitivitäten waren jeweils die sensitivsten Konfigurationen für die beiden Sensoren, um möglichst kleine Änderungen wahrnehmen zu können. Zur Rekonstruktion der realen Gyroskopwerte wurden die gelieferten Werte, nach der Formel 5.2, in rad/s umgerechnet und über eine Multiplikation mit $250°/s$ und eine Division durch den aufgespannten Wertebereich von Short-Werten, auf den passenden Wertebereich für die Winkelgeschwindigkeit skaliert. Die Magnetometerwerte mussten ebenfalls skaliert werden, aber hatten dafür feste Skalierwerte, welche bei der Kalibrierung festgelegt wurden und sich über bestimmte Register der IMUs auslesen ließen [Mpub]. Für die IMU am Finger waren dies beispielsweise folgende Werte

Datensatz	Bezeichnung	Vektorgröße
Finger	Touch alleine	20
	Touch- und IMU_{Finger}	80
	Touch- und $IMU_{Handgelenk}$	80
	Touch- und IMU_{Finger} und $IMU_{Handgelenk}$	140
	Touch- und IMU_{Finger} und $IMU_{Handgelenk}$ und Magnetometern	200
Stylus	Touch alleine	20
	Touch- und IMU_{Stylus}	80
	Touch- und IMU_{Finger} und IMU_{Stylus}	140
	Touch- und $IMU_{Handgelenk}$ und IMU_{Stylus}	140
	Touch- und IMU_{Finger} und $IMU_{Handgelenk}$ und IMU_{Stylus}	200

Tabelle 5.1: Die Tabelle zeigt die zehn verschiedenen Modelle mit dem zugehörigen Datensätzen, aus welchem die zum Training verwendeten Daten stammen, der genauen Modellbezeichnung, welche die verwendeten Komponenten enthält und der Größe des zugehörigen Merkmalsvektors.

für die x-, y- und z-Achsenwerte: 0,1798096, 0,1798096 und 0,1733669. Die internen Sensordaten wurden vom Android-System bereits in skalierter Form geliefert, also befanden sich anschließend alle Sensordaten im gleichen Wertebereich.

Da keine Sensordaten vorhergesagt werden sollten, musste lediglich der 20 Werte große Merkmalsvektor für die Eingabedaten mit zusätzlichen Sensorwerten ergänzt werden, welche je mit ihren x-, y- und z-Werten in dieser Reihenfolge an den Vektor angehängt wurden. In den hier aufgelisteten Modellen wurden nur die IMU-Daten verwendet, die dem Touchpunkt 11 jeder Trajektorie zugeordnet wurden. Der Punkt 11 stellt den letzten bekannten Touchpunkt einer aktiven Vorhersage dar, da alle folgenden Punkte als Ground-Truth-Punkte deklariert wurden. Der Verlauf der Werte der letzten zehn IMU-Pakete ist somit der aktuellste IMU-Datenverlauf und macht die vorherigen IMU-Pakete überflüssig. Von jeder IMU wurden in den folgenden Modellen stets nur die dreiachsigen Accelerometer- und Gyroskopdaten verwendet, da vorangegangene Testtrainings mit Magnetometerdaten stets schlechtere Ergebnisse brachten. Die Hinzunahme einer IMU zum Merkmalsvektor entspricht also der Hinzunahme von weiteren 60 Werten zu den Eingabedaten (zehn Pakete von Touchpunkt 11, mal zwei Sensoren, mal drei Achsen = 60 Werte). Lediglich das fünfte Modell verwendet Magnetometerdaten, um einen späteren Vergleich zu ermöglichen. Die ersten fünf Modelle wurden alle aus den Daten erstellt, welche durch Fingerbenutzung in der Studie zur Datenerhebung gesammelt wurden, die letzteren fünf durch die entsprechenden Stylusdaten der Studie. Die internen Sensordaten wurden für diese Modelle nicht verwendet, da sie in vorangegangenen Testtrainings keine Verbesserungen zeigten und die Größe des Merkmalsvektors erhöht hätten. Auf diese Weise wurden zehn verschiedene Merkmalsvektoren unterschiedlicher Größe erstellt, mit welchen anschließend zehn Modelle trainiert wurden. Sie werden in Tabelle 5.1 benannt.

5.3 Training

Beim Training der neuronalen Netze wurden mehrschichtige „feedforward“-Netze trainiert, welche auf den Eingabedaten der Merkmalsvektoren die rotierten Verschiebungsvektoren der Ground-Truth Touchdaten von Finger- oder Stylusbewegungen durch Regression vorhersagen [TH11; Tea17]. Diese Ergebnisse können anschließend über entsprechende Gegenrechnungen wieder in Touchpunktkoordinaten zurückgerechnet werden. Die „feedforward“-Netze beschreiben dabei künstliche neuronale Netze, in welchen Daten nur an die nächste Schicht gereicht werden können. Es gibt demnach keine Zyklen im entstehenden Graph. Durch diese Limitation handelt es sich um relativ einfache neuronale Netze [TH11].

Allgemein bestehen solche Netze aus mehreren Schichten, mit je einer fest definierten Anzahl Knoten, auch Neuronen genannt. Diese sind in Schichten angeordnet und Neuronen einer Schicht sind mit den Neuronen benachbarter Schichten verbunden. Jedes Neuron im neuronalen Netz hat mehrere Eingänge und einen Ausgang, über welche Werte transportiert werden. Die eingehenden Werte eines Neurons entsprechen den ausgehenden Werten der Neuronen der vorherigen Schicht. Sie werden mit Gewichten multipliziert und zusammenaddiert. Neuronen der versteckten Schicht verfügen zusätzlich über eine Aktivierungsfunktion am Ausgang, die den Ausgangswert skaliert. Über Bias-Neuronen können die Schwellwerte der Aktivierungsfunktionen in bestimmte Wertebereiche verschoben werden, was zielorientierte Lernprozesse ermöglicht. Die Gewichte werden im Verlauf des Trainings definiert und in jedem Trainingsschritt angepasst. Für diese Form der Optimierung wird eine Kostenfunktion definiert, mit der das Netz abschätzt, wie es sich den Ground-Truth-Daten annäheren kann. Die erste Schicht besteht immer aus so vielen Neuronen, wie sich Elemente im Merkmalsvektor befinden, da sie diesen repräsentiert. Anschließend folgen optional versteckte Schichten (englisch: hidden layers), mit beliebig vielen Neuronen, welche während des Trainings mit anderen Neuronen vernetzt werden und so gewichtete Werte empfangen, verarbeiten und weitersenden. An letzter Stelle befindet sich die Ausgabeschicht, welche so viele Knoten enthält, wie die Größe des Vektors für die Ground-Truth-Daten und die finalen Ergebnisse für die Eingabedaten liefert [Nie17]. Der Lernprozess wird über Optimierer und Trainingszeiten definiert. Optimierer minimieren die Kosten durch die Kostenfunktion in jedem Schritt. Trainingszeiten werden in Epochen angegeben. Eine Epoche ist eine einmalige Anpassung der Gewichte zur Minimierung der Kosten durch den Optimierer. Ein neuronales Netz durchläuft während des Trainings mehrere Epochen, bis die Kosten durch die Anpassung der Gewichte hinreichend optimiert wurden. Die Anpassung der Gewichte läuft, in jedem Schritt, durch zwei Phasen, die Vorwärtspropagierung und die Rückwärtspropagierung. Bei der Vorwärtspropagierung werden Eingangswerte am Netz angelegt, mit den zuletzt zugewiesenen Gewichten die Ausgabewerte berechnet und mit den Sollwerten verglichen. Bei der Rückwärtspropagierung werden dann die Gewichte der Neuronen, ausgehend von den Sollwerten der Ausgabe, so angepasst, dass die Ausgabe des Netzes den Sollwerten möglichst nahe kommt.

Im Verlauf dieser Arbeit wurden viele verschiedene Konfigurationen von Aktivierungsfunktionen, Optimierern, Kostenfunktionen, Trainingszeiten, Anzahl versteckter Schichten und

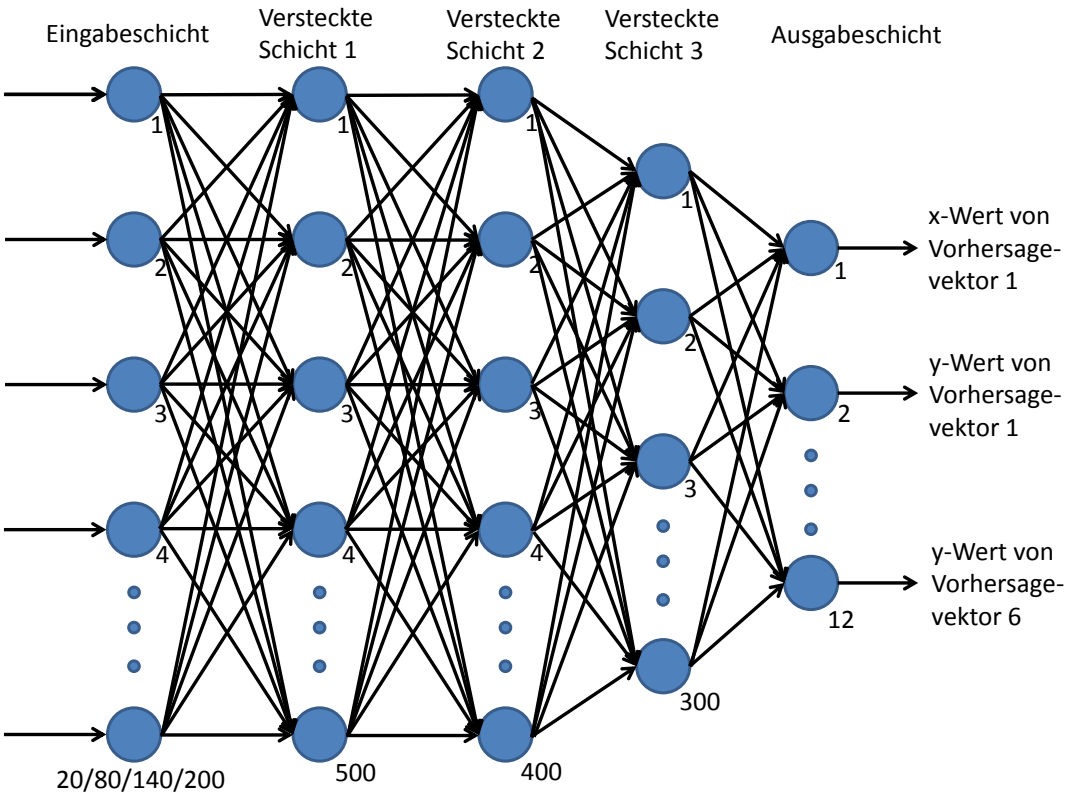


Abbildung 5.2: Die Abbildung zeigt den Aufbau der beschriebenen neuronalen Netze mit allen Schichten und Neuronen. Die Neuronenanzahl der Eingabeschicht wird je nach Modell (bzw. Merkmalsvektor) auf 20, 80, 140 oder 200 festgelegt. Die Signalrichtung ist ausschließlich von links nach rechts, da es sich um ein feedforward-Netz handelt. Die 12 Ausgabeneuronen enthalten am Ende die entsprechenden x- und y-Werte der Vorhersagevektoren.

Anzahl ihrer Neuronen getestet. Die abschließend beste Struktur wurde abhängig von den getesteten zehn Modellen ausgewählt. Die Eingabeschichtschicht (Anzahl Neuronen entspricht Elementen der Eingabe) wurde über drei versteckte Schichten mit 500, 400 und 300 Neuronen zu einer Ausgabeschicht verbunden, welche für jedes Modell exakt 12 Neuronen enthielt, mit den x- und y-Werten der rotierten Verschiebungsvektoren der zu vorhersagenden Touchbewegungen, welche zwischen 16,6ms (1. Verschiebungsvektor) und 99,9ms (6. Verschiebungsvektor) in der Zukunft lagen. Die beschriebene Struktur wurde in Abbildung 5.2 visualisiert. Als Optimierer wurde der AdaGrad Algorithmus („adaptive gradient algorithm“) verwendet. AdaGrad ist ein adaptives Gradientenverfahren, welches über den negativen Gradienten der Kosten stets den Weg der größten Minimierung wählt, bis numerisch keine Verbesserung mehr stattfindet [DHS11]. Als Aktivierungsfunktionen, der ersten zwei versteckten Schichten, wurden „rectified linear units“ (Abkürzung: ReLU) [NH10] und für die dritte versteckte Schicht eine Sigmoidfunktion [HM95] verwendet. Die Nutzung von zwei verschiedenen Aktivierungs-

funktionen sorgt für den Ausgleich der Schwächen jeder einzelnen Aktivierungsfunktion, in bestimmten Situationen [Tea17]. Für die Kostenfunktion wurde die euklidische Distanz, zwischen den x- und y-Werten der vorhergesagten rotierten Verschiebungsvektoren v und den realen Werten der rotierten Verschiebungsvektoren der Ground-Truth-Daten r, über alle Vorhersagezeiten $t \in T$, herangezogen, welche in der Gleichung 5.3 dargestellt ist.

$$Kosten = \sqrt{\frac{\sum_t^T (v_t - r_t)^2}{|T|}} \tag{5.3}$$

Für die Initialisierung der Gewichte wurde die Xavier Initialisierung von Glorot und Bengio verwendet [GB10]. Sie eignet sich gut dafür, die Gewichte des Netzes randomisiert so zu initialisieren, dass diese in einem geeigneten Bereich liegen [Yaz16]. Mit der beschriebenen Struktur wurden die jeweiligen neuronalen Netze dann so viele Epochen lange trainiert, bis die Änderungsrate der Kosten geringer als 0,001 war. Das geschah nach durchschnittlich 660,5 Trainingsepochen (SD = 86,7).

Zum späteren Vergleich der Modelle wurde eine Leave-One-Out-Kreuzvalidierung implementiert [KR99], welche das neuronale Netz für ein Modell 18 Mal trainierte, wobei jedes Mal die Daten eines anderen der 18 Teilnehmer nicht mit ins Training einflossen. Die ausgelassenen Daten dienten je in jedem der 18 Durchläufe als Testdaten, um die Genauigkeit des jeweiligen Modells zu testen. Dafür wurden die euklidischen Distanzen in Pixeln, zwischen den aus den Vorhersagen resultierenden, absoluten Koordinaten (durch Rückgängigmachen der Rotation und Subtraktionen) der vorhergesagten Touchpunkte und den realen Koordinaten der Trajektorien des ausgelassenen Benutzers, berechnet und dokumentiert. Auf diese Weise konnten personenunabhängige Vergleichsdaten erhoben werden, welche die Genauigkeit des Modells auf neuen Eingaben neuer Nutzer beschreiben, auf denen es nicht trainiert wurde, was einen späteren Vergleich der Modelle erlaubte. Die berechneten euklidischen Pixeldifferenzen wurden zusätzlich für jede der drei einzelnen Aufgaben aus der ersten Studie erhoben, damit der Genauigkeitsunterschied der Modelle bei den einzelnen Aufgaben verglichen werden konnte. Die gewählten Modelle, für die spätere Evaluationsstudie im nächsten Kapitel, wurden vor dem Einsatz noch einmal mit den Daten aller Nutzer trainiert, um mehr Trainingsreferenzdaten zu verinnerlichen.

5.4 Ergebnisse

Die euklidischen Pixeldifferenzen, der vorhergesagten Punkte und der realen Touchpunkte aus der Leave-One-Out-Kreuzvalidierung, dienten als Hauptanhaltspunkt zum Vergleich der Modelle. Auf das Google Nexus 7 bezogen, bedeutet ein Pixel etwa eine reale Distanz von 0,08mm. Die Modelle wurden einmal im Hinblick auf die Gesamtgenauigkeit und einmal auf die Genauigkeit in Bezug auf die einzelnen Aufgaben verglichen. Alle Distanzen waren dabei, aufgrund der Leave-One-Out-Kreuzvalidierung, nutzerunabhängig.

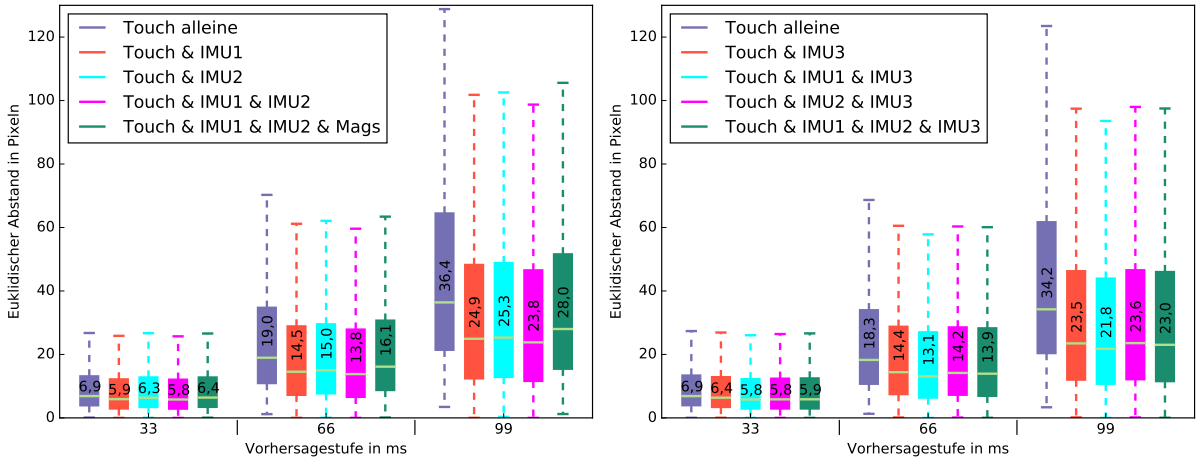


Abbildung 5.3: Die abgebildeten Grafiken stellen die Boxplots der euklidischen Distanzen der Vorhersagen zu den realen Touchpositionen, der zehn Modelle, abhängig von den verschiedenen Vorhersagestufen 33ms, 66ms, und 99ms (gerundete Werte), über alle Aufgaben dar. Die linken Boxplots gehören zu den Finger-Modellen, die rechten zu den Stylus-Modellen. Die IMU-Bezeichnungen wurden abgekürzt (IMU1 = IMU_{Finger}, IMU2 = $IMU_{Handgelenk}$, IMU3 = IMU_{Stylus}, Mags = Magnetometer). Die Werte in den Boxen repräsentieren den Bereich, den die mittleren 50% der Distanzen aufspannen, die Fühler spannen den Bereich der jeweils davor liegenden 22,5% der Daten auf. Die Linien in den Boxen entsprechen dem Median, welcher innerhalb der Box zusätzlich als Zahlenwert annotiert wurde.

Der erste Vergleich der Modelle, über die Distanzen, über alle drei Aufgaben hinweg, zeigte, dass alle IMU-Modelle im Durchschnitt niedrigere Distanzen, demnach mehr Genauigkeit aufwiesen, als die Modelle, welche nur auf Touchdaten trainiert wurden, wie in Abbildung 5.3 dargestellt. Auch die Fühler im entsprechenden Graph sind bei den IMU-Modellen kürzer, was für weniger hohe Abweichungen und Ausreißer spricht. Das genauste Modell aus den Fingerdaten ist das, welches zusätzlich zu den Touchdaten die zwei IMUs IMU_{Finger} (IMU am Finger) und $IMU_{Handgelenk}$ (IMU am Handgelenk) nutzt. Dieses hat durchschnittliche euklidische Distanzen von 10,32 Pixeln (SD = 15,14 Pixel) bei einer 33ms Vorhersage, 22,67 Pixeln (SD = 29,10 Pixel) bei einer 66ms und 37,30 Pixeln (SD = 44,45 Pixel) bei einer 99ms Vorhersage. Das entsprechende IMU-lose Modell, welches auf denselben Touchdaten trainiert wurde, weist 11,41 Pixel (SD = 15,39 Pixel), 28,96 Pixel (SD = 32,84 Pixel) und 53,07 Pixel (54,78 Pixel) Distanz für die gleichen Vorhersagestufen auf. Im Vergleich ist das Modell mit den zwei IMUs durchschnittlich um 9,55% bei einer 33ms, 21,71% bei einer 66ms und 29,72% bei einer 99ms Vorhersage genauer, als das entsprechende Modell ohne IMUs für Fingereingaben. Die Hinzunahme von Magnetometerdaten zum Training verschlechterte die Genauigkeit des Modells mit den beiden IMUs IMU_{Finger} und $IMU_{Handgelenk}$. Dadurch ergaben sich Distanzen von 11,02 (SD = 15,49), 25,27 (SD = 29,90) und 42,01 Pixeln (SD = 45,80) für die drei Vorhersagestufen,

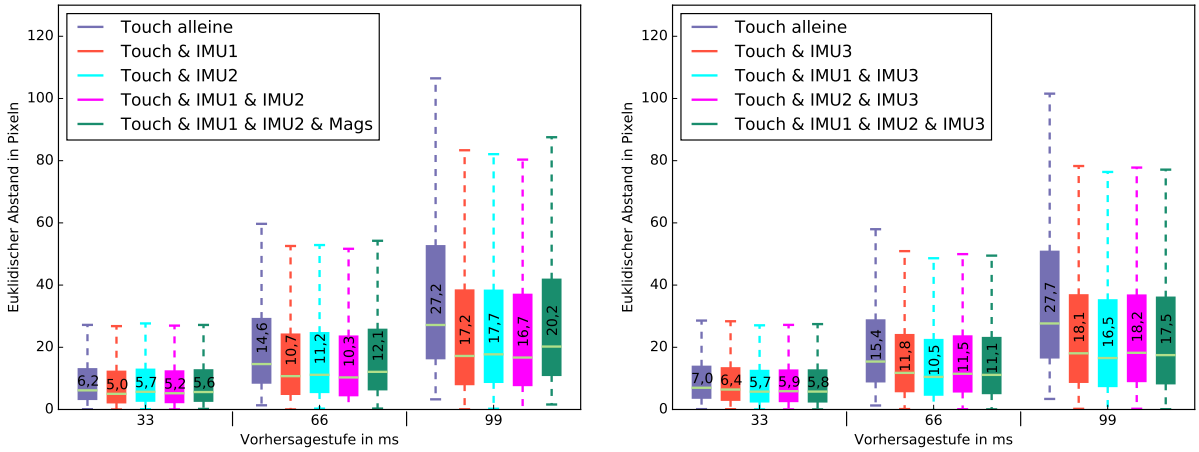
	Dist. 33ms (px)		Dist. 66ms (px)		Dist. 99ms (px)	
Modell	Durchs.	SD	Durchs.	SD	Durchs.	SD
Touch alleine	11,41	15,39	28,96	32,84	53,07	54,78
Touch + IMU1	10,42	15,28	23,56	29,61	38,77	45,63
Touch + IMU2	10,94	15,33	24,14	29,79	39,39	45,92
Touch + IMU1 + IMU2	10,32	15,14	22,67	29,10	37,30	44,45
Touch + IMU1 + IMU2 + Mags	11,02	15,49	25,27	29,90	42,01	45,80
Touch alleine	11,99	16,45	29,35	34,99	52,39	57,53
Touch + IMU3	11,52	16,65	24,50	32,07	38,69	47,97
Touch + IMU1 + IMU3	10,89	16,28	22,97	31,17	36,51	46,44
Touch + IMU2 + IMU3	10,98	16,36	24,24	31,71	38,80	47,79
Touch + IMU1 + IMU2 + IMU3	11,03	16,52	24,01	31,97	38,38	48,16

Tabelle 5.2: In der Tabelle wurden die Durchschnittswerte der euklidischen Distanzen in Pixeln mit Standardabweichungen (SD) für alle zehn Modelle dargestellt. Die Finger- und Stylusmodelle wurden in der Mitte räumlich getrennt (Fingermodelle oben, Stylusmodelle unten). IMU-Bezeichnungen wurden abgekürzt (IMU1 = IMU_{Finger}, IMU2 = $IMU_{Handgelenk}$, IMU3 = IMU_{Stylus}, Mags = Magnetometer).

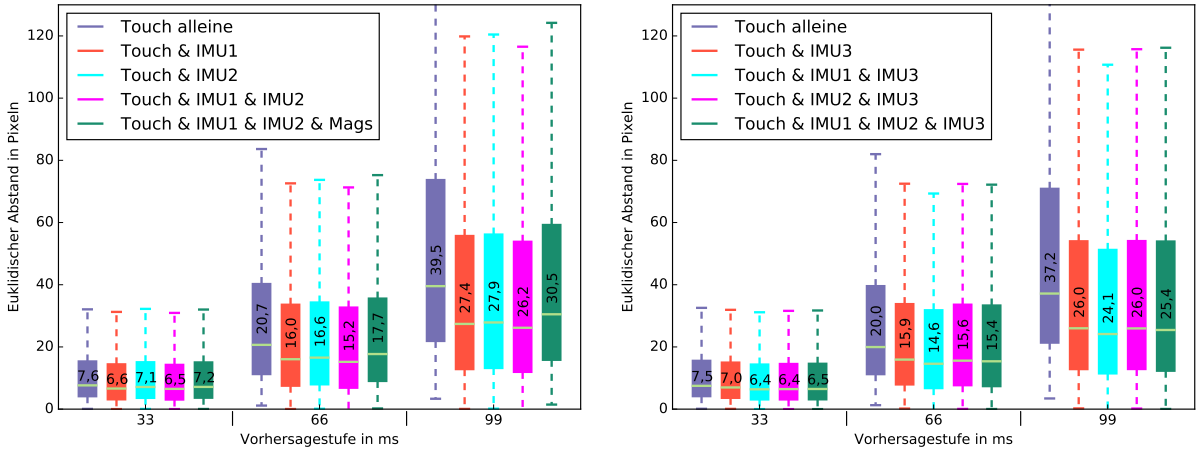
was eine Verschlechterung der Genauigkeit von 6,35%, 10,29% und 11,21% gegenüber dem Modell ohne Magnetometer bedeutet.

Für Stylusdaten ergaben sich ähnlich Ergebnisse. Am besten schnitt hier das Modell mit den IMUs IMU_{Finger} und IMU_{Stylus} (IMU am Stylus) ab. Dieses hat durchschnittliche euklidische Distanzen von 10,89 Pixeln bei einer 33ms Vorhersage (SD = 16,28 Pixel), 22,97 Pixeln bei einer 66ms (SD = 31,17 Pixel) und 36,51 Pixeln bei einer 99ms Vorhersage (SD = 46,44 Pixel), auf den Testdaten. Das entsprechende Modell, basierend auf den Touchdaten der Styluseingaben ohne IMUs, erzeugte hingegen Distanzen von 11,99 (SD = 16,45), 29,35 (SD = 34,99) und 52,39 Pixeln (SD = 57,53), für die drei Vorhersagestufen. Damit ergeben sich durchschnittliche Genauigkeitsverbesserungen von entsprechend 9,17%, 21,74% und 30,31%, durch die Verwendung der zwei IMUs mit dem Stylus. Die Hinzunahme der dritten IMU zum Modell mit den beiden IMUs IMU_{Finger} und IMU_{Stylus} verbesserte die Genauigkeit nicht weiter. Die Modelle, welche nur eine IMU nutzen, zum Beispiel das IMU_{Stylus} Modell für Styluseingaben und das $IMU_{Handgelenk}$ Modell für Fingereingaben, sind gegenüber den besten Modellen ihrer Eingabemethoden durchschnittlich nur um höchstens einen Pixel bei den 33ms Vorhersagen und höchstens zwei Pixel bei den 66ms und 99ms Vorhersagen ungenauer. In Tabelle 5.2 wurden alle Durchschnittsdistanzen mit Standardabweichung für alle zehn Modelle dargestellt, um weitere Vergleiche zu erlauben.

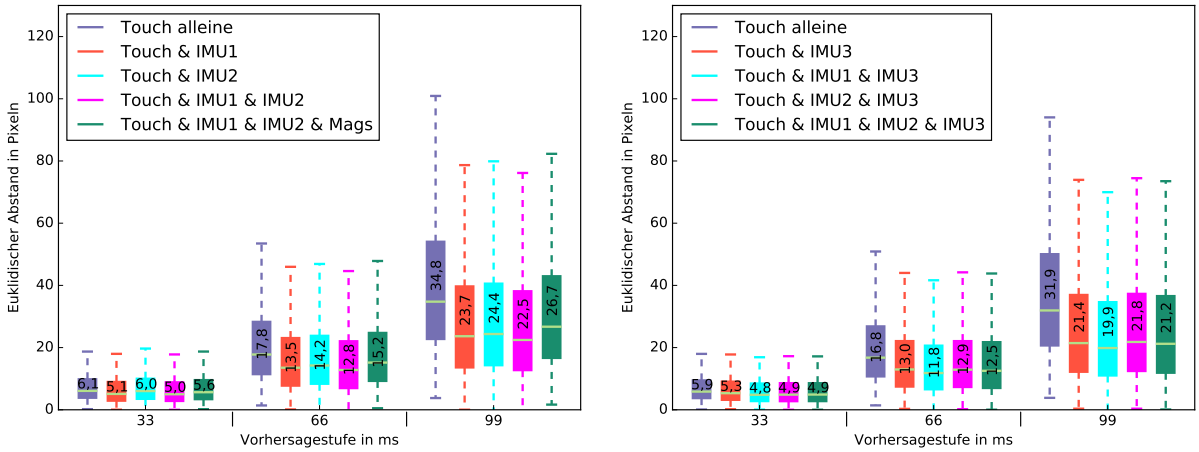
Die Durchschnittswerte sagen wenig über die Genauigkeit der Modelle bei den einzelnen Aufgaben aus. Außerdem wurden für jede Aufgabe während der ersten Studie unterschiedlich viele Daten erzeugt, was bedeutet, dass eine schlechte Aufgabe den Gesamtschnitt eines Modells verschlechtert. Um die Genauigkeit der Modelle bei den einzelnen Aufgaben zu untersuchen,



(a) Distanzen bei Fitts' Law



(b) Distanzen beim Zeichnen



(c) Distanzen beim Schreiben

Abbildung 5.4: Die Grafiken zeigen Boxplots der euklidischen Durchschnittsdistanzen für die einzelnen Aufgaben. Die linken Grafiken zeigen jeweils die Finger- und die rechten Grafiken die Stylusmodelle. Der Aufbau der Boxplots entspricht dem der vorherigen Boxplots. Abkürzungen: IMU1 = IMU_{Finger}, IMU2 = $IMU_{Handgelenk}$, IMU3 = IMU_{Stylus}, Mags = Magnetometer

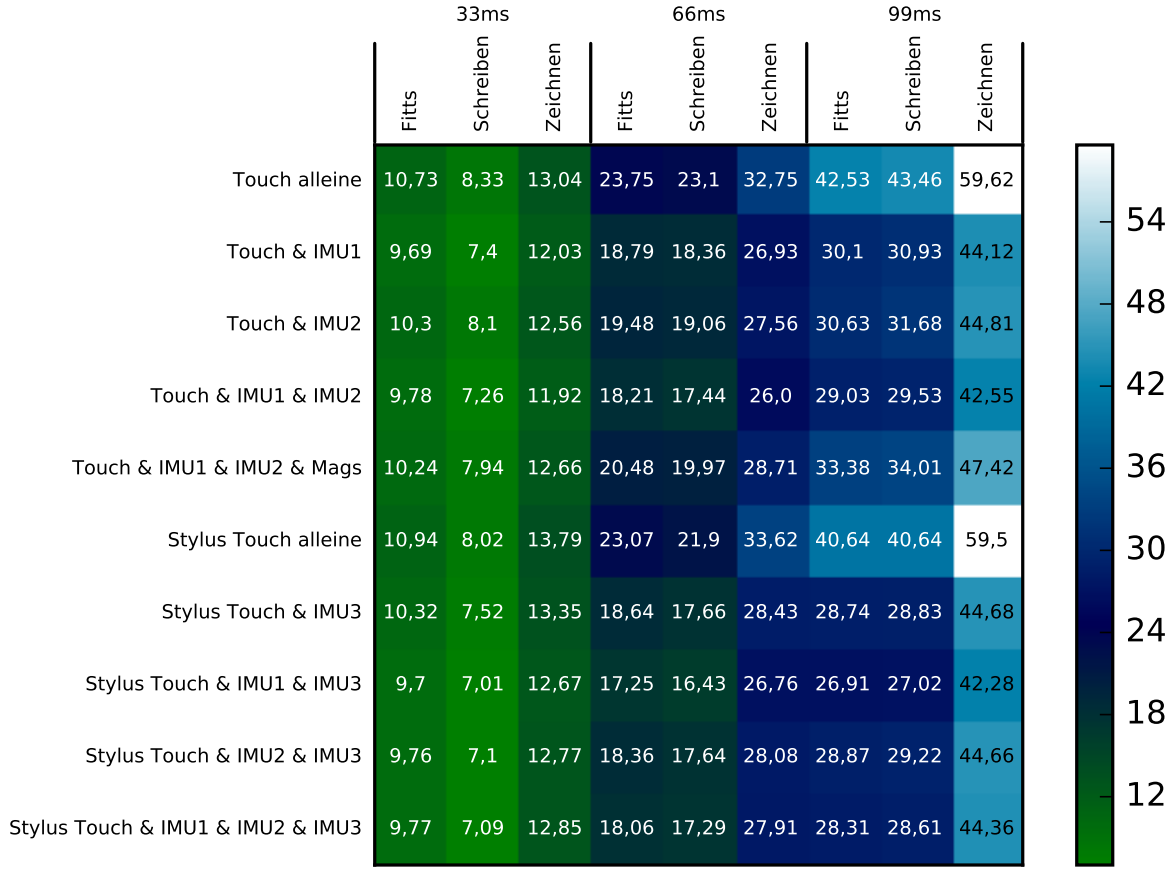


Abbildung 5.5: Die Matrix enthält die Durchschnittswerte in Pixeln der gerundeten euklidischen Distanzen der jeweiligen Modelle (Zeilen), für die jeweiligen Aufgaben und Vorhersagestufen (Spalten). Zur Hervorhebung der Unterschiede sind die einzelnen Felder je nach Zahlenwert eingefärbt. Je niedriger die Zahlenwerte (grün), desto genauer sind die Vorhersagen für diese Vorhersageart. IMU-Bezeichnungen wurden abgekürzt (IMU1 = IMU_{Finger}, IMU2 = $IMU_{Handgelenk}$, IMU3 = IMU_{Stylus}, Mags = Magnetometer).

wurden in 5.4 Boxplots für die einzelnen Aufgaben erstellt. Zu erkennen ist, dass die Verhältnisse zwischen den Modellen für die Einzelaufgaben auch den Verhältnissen der Modelle für die zusammengefassten Zahlen aller Aufgaben entsprechen. Die besten Modelle über alle Durchschnittswerte sind demnach auch die besten Modelle für die jeweiligen Einzelaufgaben, mit nur wenigen Ausnahmen und die Modelle ohne IMUs sind die schlechtesten Modelle für jede Aufgabe, bezogen auf die euklidischen Distanzen der Vorhersagen, dargestellt in den Boxplots. Dies sieht man noch deutlicher an den Durchschnittswerten, welche in Abbildung 5.5 dargestellt sind. Das Modell Touch und IMU_{Finger} ist beispielsweise bei der 33ms Vorhersage im Schnitt um 0,09 Pixel genauer, als das sonst bessere Modell Touch, IMU_{Finger} und $IMU_{Handgelenk}$, welches dafür bei allen anderen Aufgaben und Vorhersagestufen genauer ist.

Klarer sind die Unterschiede zwischen den einzelnen Aufgabentypen zu erkennen. Bei den 33ms Vorhersagen sind die durchschnittlichen euklidischen Distanzen bei den Schreibaufgaben für jedes Modell am geringsten (Durchschnitt der Pixeldistanzen beim Schreiben = 7,58 Pixel). Die Distanzen für die Fitts' Law-Aufgaben sind nur um durchschnittlich 2,54 Pixel höher (Durchschnitt = 10,12 Pixel), die Zeichendistanzen dagegen sind durchschnittlich um 5,18 Pixel höher (Durchschnitt = 12,76 Pixel), als die der Schreibaufgaben. Bei den 66ms und 99ms Vorhersagen gleichen sich die Fitts' Law- (66ms Durchschnitt = 19,61 Pixel, 99ms Durchschnitt = 31,91 Pixel) und Schreibdistanzen (66ms Durchschnitt = 18,89 Pixel, 99ms Durchschnitt = 32,29 Pixel) noch weiter an, mit durchschnittlichen Unterschieden von 0,77 Pixeln für 66ms und 0,38 Pixeln für 99ms Vorhersagen. Die Abweichungen der Zeichenaufgabendistanzen (66ms Durchschnitt = 28,68 Pixel, 99ms Durchschnitt = 47,40 Pixel) zu denen der Schreibaufgaben wird dagegen größer, mit durchschnittlich um 9,79 Pixel bei 66ms Vorhersagen und um 15,11 Pixel bei 99ms Vorhersagen größeren Distanzen. Für das Modell der Stylusdaten, welches ohne IMU trainiert wurde, gibt es einen Distanzunterschied von 18,86 Pixeln zwischen Durchschnittsdistanzen bei Zeichenaufgaben (Durchschnitt = 59,5 Pixel) und den anderen beiden Aufgaben (Durchschnitt bei Fitts' Law und Schreiben = 40,64), bei einer 99ms Vorhersage. Bei IMU-Modellen ist dieser Unterschied auch teilweise über 15 Pixel groß, wie bei dem Modell Touch und IMU_{Finger} und IMU_{Stylus} mit durchschnittlich um 15,26 Pixel größeren Distanzen von Zeichenvorhersagen gegenüber Schreibvorhersagen, bei 99ms Vorhersagen. Auch bei den 66ms Vorhersagen ist dieser Unterschied deutlich zu betrachten.

Zwei dreifaktorielle ANOVAs (ANOVA bezeichnet eine Varianzanalyse), eine für die Distanzen der Fingermodelle und eine für die Distanzen der Stylusmodelle, mit der Vorhersagezeit, dem Aufgabentyp und der IMU-Nutzung als unabhängige Variablen, wurden auf den Daten durchgeführt. Es wurden dabei multiple paarweise Vergleiche, unter der Verwendung der Bonferroni-Methode, durchgeführt, welche dafür sorgt, dass durch multiple Vergleiche entstehende Fehler neutralisiert werden [Abd07]. Zwischen allen Modellen und Werten der unabhängigen Variablen konnten auf diese Weise statistisch signifikante Unterschiede gefunden werden (p-Wert < 0,001; der p-Wert entspricht einer Signifikanzstufe und gibt das Risiko an, dass fälschlicherweise ein Unterschied gefunden wird, welcher real nicht existiert; im Folgenden Text wird er mit p abgekürzt), außer zwischen den auf Stylusdaten basierenden Modellen Touch & IMU_{Finger} & IMU_{Stylus} und Touch & $IMU_{Handgelenk}$ & IMU_{Stylus} ($p < 0{,}996$).

5.4.1 Diskussion

Im Verlauf der Erstellung der neuronalen Netze wurde mit unterschiedlichen Strukturen gearbeitet. Die beschriebene Struktur entspricht der besten Architektur, welche durch manuelle Suche gefunden wurde. Mehr Forschung in diesem Bereich wäre aber denkbar. Es wurden auch Modelle trainiert, welche die internen Sensordaten des Tablets nutzten. Diese Modelle wiesen aber entweder schlechtere Werte auf, als die Modelle ohne Nutzung interner Sensoren, oder zeigten keine Verbesserungen. Eine mögliche Erklärung ist, dass die besagten Sensorwerte sehr unregelmäßig geliefert wurden, nämlich nur dann, wenn die Hardware Änderungen

wahrnahm und nicht kontinuierlich, wie die IMU-Daten. Durch die gezwungene feste Anzahl von Eingabeneuronen geraten so auch interne Sensorwerte zusammen ins Training, die einen zu großen Abstand zueinander haben, als dass das neuronale Netz aus ihnen lernen kann. Zur zukünftigen Umgehung dieses Problems könnte man versuchen, die internen Sensoren, vor einer Datenerhebungsstudie, dazu zu zwingen, kontinuierlich Daten zu senden, indem man beispielsweise virtuelle Änderungsereignisse erstellt. Außerdem wäre eine Verknüpfung der dreidimensionalen Räume der IMU-Daten und der internen Sensordaten hilfreich, um dem neuronalen Netz zu erleichtern, die Zusammenhänge zwischen diesen Räumen und der zweidimensionalen Ebene des Touchscreens zu finden. Die hier trainierten Netze haben den Zusammenhang zwischen IMU-Daten und Bildschirm auch ohne die internen Sensordaten gelernt, was daran liegen könnte, dass fast alle Teilnehmer der Studie zur Datenerhebung das Tablet, während der Aufzeichnungen, flach auf dem Tisch liegen hatten. Interne Sensorwerte könnten demnach auch andere Interaktionen, zum Beispiel Touchinteraktion im Stehen, mit gehaltenem Tablet, ermöglichen und präziser machen, als mit den IMUs alleine.

Auch die Magnetometerdaten waren beim Training der neuronalen Netze nicht sehr hilfreich. Sie führten fast bei jedem getesteten Modell zu Verschlechterungen der errechneten Durchschnittsdistanzen. Das wiederum lässt sich auf ihre Fehleranfälligkeit zurückführen und auf die wesentlich geringere Aktualisierungsrate, verglichen mit den anderen Sensoren. In anderen Aufbauten könnte man Magnetometer zusätzlich abschirmen, um sie vor Störeinflüssen zu schützen. Das oben vorgestellte Modell Touch & $\text{IMU}_{\text{Finger}}$ & $\text{IMU}_{\text{Handgelenk}}$ & Magnetometer ist das Beste der trainierten Modelle, welche mit Magnetometerdaten trainiert wurden, da bei anderen die Distanzen noch höher waren. Das kann daran liegen, dass die Nutzung von zwei Magnetometern, in der Finger- und Handgelenk-IMU, dafür sorgen, dass die Fehler eines Magnetometers, beim Training der neuronalen Netze, durch die Werte des zweiten Magnetometers kompensiert werden. Trotzdem ist auch dieses Modell schlechter, als dasselbe Modell ohne Magnetometer und wurde nur für Vergleiche herangezogen.

Die erstellten neuronalen Netze konnten für die Modelle mit IMUs eine Genauigkeitsverbesserung von bis knapp über 30% erreichen, gegenüber Vorhersagemodellen, die nur auf Touchdaten beruhen, wie das Modell von Henze et al. [HFS16]. Je höher die Vorhersagestufe, desto größer fällt die Verbesserung durch die IMUs aus. Die Hinzunahme einer zweiten IMU fällt aber geringer aus und befindet sich nur in einem Bereich von höchstens zwei Pixeln im Durchschnitt. Die Hinzunahme einer dritten IMU hat keine weitere Verbesserung ergeben. Vergleicht man die Durchschnittswerte der hier trainierten IMU-losen Modelle mit denen von Henze et al., so lässt sich feststellen, dass die Werte für Fitts' Law- und Zeichenaufgaben ähnlich sind. Das spricht dafür, dass deren Ansatz nachgebildet werden und durch IMUs verbessert werden konnte. Die Distanzen der Zeichenaufgabe sind in beiden Ansätzen und jedem Modell am höchsten. Das war zu erwarten, da die Zeichenbewegungen, im Gegensatz zu Schreibbewegungen und Trajektorien bei Fitts' Law-Aufgaben, die einem wohldefinierten Muster folgen, sehr willkürlich sein können, weshalb die neuronalen Netze Schwierigkeiten haben, diese zu generalisieren. Viele Beispiele haben hingegen bereits gezeigt, dass das Erlernen von

Buchstabenmustern, aufgrund ihres Informationsgehalts, für neuronale Netze sehr gut möglich ist[2].

Ein Unterschied zu der Arbeit von Henze et al. lässt sich in den durchschnittlichen Pixeldistanzen für Schreibaufgaben finden, welche dort signifikant geringer sind, als die, der hier vorgestellten neuronalen Netze. Das hat den Grund, dass die dort verwendete Android-Applikation, zum Sammeln der Daten, im Google Play Store vertrieben wurde. Unter anderem konnten Nutzer mit der Applikation handschriftliche Notizen machen, welche sie dann Henze et al. online zur Verfügung stellen konnten [HFS16]. Auf diese Weise können viel mehr Trainingsdaten gesammelt werden. Der größere Datensatz von Henze et al. ermöglichte ihrem neuronalen Netz eine höhere Genauigkeit bei der Mustererkennung von Schreibaufgaben. Das erklärt, warum sich nur die Genauigkeiten von Schreibaufgaben bei den hier vorgestellten IMU-losen Modellen und denen von Henze et al. unterscheiden, Fitts' Law- und Zeichenaufgaben aber nicht merklich.

Ein weiteres auffälliges Merkmal, vor allem bei höheren Vorhersagestufen (66ms und 99ms), der erstellten neuronalen Netze ist, dass der Median der euklidischen Distanzen der Vorhersagen zu den realen Werten klein ist. Betrachtet man die Boxplots, so fällt auf, dass der Median bei höheren Vorhersagestufen im unteren Bereich der Boxen liegt. Die entsprechenden Durchschnittswerte sind beträchtlich höher als der Medianwert. Der Median teilt die Distanzen in zwei gleich große Teile. An den Fühlern der Boxplots lässt sich erkennen, dass die 50% der Daten, welche über dem Median liegen, einen großen Bereich aufspannen. Das deutet darauf hin, dass viele ungenaue Vorhersagen und Ausreißer die Durchschnittswerte in die Höhe treiben. Diese hohen Abweichungen lassen sich bei einer Verwendung des entsprechenden neuronalen Netzes herausfiltern, um später die Benutzerfreundlichkeit zu erhöhen.

Gesamt lässt sich sagen, dass die trainierten Modelle gute Vorhersagen treffen. Die IMU-Modelle treffen genauere Vorhersagen als die IMU-losen Modelle. Die aufgenommenen internen Sensordaten und Magnetometerdaten brachten aufgrund verschiedener Fehler keine Besserungen und die Schreibperformanz ist aufgrund der Größe der Referenzdaten etwas schlechter als es möglich wäre. Verbesserungen der Modelle sind also noch möglich. Von den verglichenen Modellen sollen nun die ausgewählt werden, welche in einer zweiten Studie evaluiert werden sollen.

5.5 Modellvergleich und -auswahl

Die Modelle wurden bereits auf die Genauigkeit ihrer Vorhersagen verglichen. Zur Auswahl der Modelle für die Evaluationsstudie sind aber auch andere Faktoren zu berücksichtigen. Zuerst muss festgelegt werden, welche Vorhersagestufen getestete werden sollen. Henze et al.

[2]Beispiel für Zeichenerkennung mit neuronalen Netzen: https://www.tensorflow.org/get_started/mnist/beginners

testeten ihre Modelle für 33ms und 66ms Vorhersagen, weil 99ms Vorhersagen, aufgrund von Ungenauigkeiten, sehr viel unerwünschten Jitter und Abweichungen enthielten [HFS16]. Die IMU-verbesserten Modelle reduzieren zwar diese Fehlverhalten für 99ms Vorhersagen, haben aber trotzdem beträchtlich mehr Ungenauigkeiten, als bei den anderen Vorhersagestufen. Deshalb und um einen späteren Vergleich der Ergebnisse der hier verwendeten Modelle mit denen von Henze et al. möglich zu machen, werden die 99ms Vorhersagen nicht weiter untersucht und weggelassen.

Es sollen vier Kriterien in der zweiten Studie untersucht werden. Zum einen der Effekt von Vorhersagen auf die Nutzerperformanz. Dafür ist es notwendig, sowohl Modelle ohne Vorhersage, als auch Modelle mit Vorhersage in der Studie zu vergleichen. Weiterhin soll der Einfluss unterschiedlicher Vorhersagestufen auf die Performanz untersucht werden. Dafür wurden bereits die zwei Stufen 33ms und 66ms ausgewählt. Weiterhin soll die Wirkung der verbesserten Genauigkeit durch die IMUs auf die Nutzerperformanz und Benutzerfreundlichkeit erforscht werden. Dafür müssen sowohl IMU-Modelle, als auch die entsprechenden IMU-losen Modelle für die ausgewählten Vorhersagestufen ausgewählt und verglichen werden. Als letztes soll auch noch der Unterschied zwischen Finger- und Stylusmodellen geklärt werden, weshalb auch hier entsprechenden Modelle gewählt werden müssen. Die vier zu untersuchenden Faktoren auf die Performanz sind resultierend Vorhersagen, Vorhersagestufen, IMUs und Stylusverwendung (Eingabemethode).

Als vergleichende Basis werden dafür zuerst Finger- und Styluseingaben ohne Vorhersage gewählt, um dem ersten Kriterium zu entsprechen. Das bedeutet ein gewähltes Modell ist ein Fingermodell ohne Vorhersage und ein zweites ein Stylusmodell ohne Vorhersage. Zur Untersuchung der IMU-Effekte werden anschließend die zwei trainierten Modelle ohne IMUs gewählt, einmal für Finger- und einmal für Stylusaufgaben, je mit 33ms und 66ms Vorhersage, um einen Vergleich mit IMU-Modellen und verschiedenen Vorhersagestufen zu ermöglichen. Es fehlen demnach zur Erfüllung aller zu untersuchenden Faktoren noch zwei IMU-Modelle, eines für Stylus- und eines für Fingereingaben mit je einer 33ms und 66ms Vorhersage.

Naheliegend ist es hier, die Modelle mit den besten Trainingsergebnissen, basierend auf einer Kreuzvalidierung, zu nehmen, also Touch & IMU_{Finger} & $IMU_{Handgelenk}$ für Fingereingaben und Touch & IMU_{Finger} & IMU_{Stylus} für Styluseingaben. Diese nutzen allerdings beide die IMU_{Finger}, welche aktuell schlecht mit kommerziellen Geräten realisiert werden kann, da dafür ein Smartring mit entsprechender Sensorik verfügbar sein muss. Solche Ringe befinden sich noch nicht im Handel, da sie noch nicht markttauglich sind, demnach gibt es keinen kommerziellen Ersatz für die IMU_{Finger}. Es gibt aber schon einige Arbeiten in diesem Bereich, die dafür sorgen könnten, dass Smartrings in Zukunft eine starke Entwicklung durchmachen und den Markt erobern [BJ17; FBL14; Mai17; Nir+15].

Stattdessen kommen die Modelle Touch & $IMU_{Handgelenk}$ für Fingereingaben und Touch & IMU_{Stylus} für Stylusaufgaben in Frage, welche nur minimal schlechtere Genauigkeit besitzen, als die ersteren zwei Modelle, dafür aber mit nur einer IMU auskommen. Das Modell Touch & $IMU_{Handgelenk}$ für Fingereingaben weicht durchschnittlich 10,94 Pixel (SD = 15,33 Pixel) für 33ms Vorhersagen und 24,13 Pixel (SD = 29,79 Pixel) für 66ms Vorhersagen von der realen

Position der Eingabe ab. Diese Werte sind um 5,7% bei 33ms und 6,1% bei 66ms schlechter, als die des besten Modells, dafür kommt das Modell ohne einen Smartring aus. Genauso das Modell mit der IMU_{Stylus} für Styluseingaben, mit durchschnittlichen Distanzen von 11,52 Pixeln (SD = 16,65 Pixel) für 33ms und 24,50 Pixeln (SD = 32,07 Pixel) für 66ms Vorhersagen. Diese Werte sind entsprechend um 5,5% und 6,2% schlechter, als die des besten Stylusmodells, was aufgrund der zeitnäheren Realisierbarkeit mit kommerziellen Geräten zu verkraften ist.

Das Modell mit der $IMU_{Handgelenk}$ könnte bereits über eine kommerzielle Smartwatch realisiert werden. Deren IMUs sind aktuell aber über die API auf niedrigere Frequenzen limitiert, oft 100Hz, weil diese für typische IMU-Aufgaben, wie etwa Bildschirmorientierungserkennung oder Bewegungsverfolgung für Fitnesstracker, vollkommen ausreichen [LXH16]. Laput et al. haben aber gezeigt, dass es möglich ist, die Daten von Smartwatch-IMUs auch hochfrequent abzufragen [LXH16]. Sie modifizieren dafür den Kernel des Android-Operationssystems einer LG G W100 Smartwatch derart, dass diese die Accelerometerdaten mit 4kHz sendet. Das würde ausreichen, um die hier vorgestellten Latenzreduzierungsverfahren mit dem $IMU_{Handgelenk}$-Modell in einem kommerziellen Gerät bereits jetzt zu verwenden.

Das Modell mit der IMU_{Stylus} könnte ebenfalls leichter realisiert werden, indem ein Stylus mit einer IMU ausgestattet wird. Es könnten auch Styli oder Smartpens verwendet werden, welche bereits mit einer IMU ausgestattet wurden, wie beispielsweise der NoteOn Smartpen[3]. Diese Modelle (Touch & $IMU_{Handgelenk}$ für Fingereingaben und Touch & IMU_{Stylus} für Styluseingaben) wurden aufgrund aller Kriterien ausgewählt und sollen in Bezug auf Performanz und Benutzerfreundlichkeit verglichen werden. Um eine Kreuzvalidierung nach allen Kriterien zu erlauben werden die Modelle ohne Vorhersage zweimal verwendet, damit alle Vorhersagestufen (0ms, 33ms und 66ms) vier Mal verwendet wurden. Zur Vereinfachung werden für die folgende Evaluierung keine neuen Modelle mit 0ms Vorhersage erzeugt, sondern die ausgewählten Vorhersagemodelle ohne Vorhersage ausgeführt.

Insgesamt werden also vier Modelle mit jeweils 0ms, 33ms und 66ms Vorhersage in der folgenden Studie verwendet. Es handelt sich dabei um die zwei Modelle, welche nur auf Toucheingaben basieren, einmal für Fingereingaben und einmal für Stylusaufgaben. Hinzu kommen die eben beschriebenen IMU-Modelle Touch & IMU_{Stylus} für Styluseingaben und Touch & $IMU_{Handgelenk}$ für Fingereingaben. Die vier Modelle werden also in der folgenden Evaluierung pro Studienteilnehmer drei Mal (für jede der drei Vorhersagestufen) ausgeführt werden, um alle Faktoren vergleichen zu können, was in zwölf Aufgabendurchläufen pro Studienteilnehmer resultiert. Vor Benutzung werden alle Modelle noch auf den kompletten Trainingsdaten trainiert, wie im Trainingsabschnitt beschrieben. Vor dem Durchführen der Studie müssen die neuronalen Netze noch entsprechend in die Android-Applikation eingebunden werden, was im folgenden Absatz beschrieben wird.

[3]Herstellerwebsite: https://hackaday.io/project/2678-noteon-smartpen

5.6 Einbindung in die Applikation

Vor der Benutzung der Modelle müssen deren neuronale Netze in die Android-Applikation auf dem Google Nexus 7 Tablet eingebunden werden. Damit dies ressourceneffizient durchgeführt werden kann, werden die Graphen der Netze über „Protocol Buffers (pb)"-Werkzeuge[4], welche TensorFlow zur Verfügung stellt, zu binären pb-Dateien zusammengefügt. Diese sind meist um das Hundertfache kleiner als die textbasierten Speicherverfahren der Graphen und weisen bessere Zugriffszeiten auf. Um auf die Graphen zugreifen zu können, werden Schnittstellenmethoden mit der Hilfe von TensorFlow Mobile[5] definiert, welche den Graphen zum Beispiel mit Eingaben versorgen und die Ergebnisse auslesen. Diese Schnittstelle liegt anschließend in C-Code vor, welcher nicht nativ auf Android ausgeführt werden kann. Damit die pb-Graphen in der Android-Applikation geladen werden können, muss zuerst die C-Schnittstelle geladen werden. Dies lässt sich auf Android über das „Java Native Interface" (JNI)[6] bewerkstelligen, welches in TensorFlow Mobile bereits enthalten ist. Das ist eine Java-Schnittstelle und Bibliothek, welche es erlaubt, C-Code auch in Java- und Android-Umgebungen auszuführen. In der C-Schnittstelle wird die Eingabegröße der Vektoren, welche als Eingabe für die neuronalen Netze dienen, in einer Methode festgelegt. Da die ausgewählten Modelle hier aber verschiedene Eingabegrößen haben, müssen mehrere solcher Schnittstellen kompiliert werden, mit den passenden Eingabegrößen, die dann zur Laufzeit, nach der Wahl des Modells, geladen werden. Für die vier gewählten Modelle werden demnach zwei Schnittstellen benötigt, eine mit einer Eingabegröße von 20 Werten, für die zwei reinen Touchmodelle und eine mit 80 Werten, für die zwei Modelle mit jeweils einer IMU. Die Schnittstellen werden dann als Bibliothek, zusammen mit den pb-Graphen, auf dem Tablet hinterlegt und können von den JNI-Funktionen von Java/Android geladen werden.

Zur Nutzung der Modelle in der Applikation müssen dann die passenden Eingabewerte an die neuronalen Netze transportiert und die Ausgabewerte abgelesen werden. Dies geschieht mit Hilfe der eben beschriebenen Strukturen. Die Eingabewerte, Touch- und IMU-Daten je nach Modell, müssen aber, wie vor dem Training der Modelle, vorverarbeitet werden. Dafür wurde eine Vorhersageroutine in der Android-Applikation erstellt, welche die Differenzen der Touchdaten berechnet, den Rotationswinkel bestimmt und die Differenzen rotiert, wie in Kapitel 5.1 beschrieben. Der Winkel der berechneten Rotation wird dabei gespeichert, damit nach der Vorhersage die Rotation rückgängig gemacht werden kann. Das geschieht über eine Rotation der Vorhersageergebnisse im Uhrzeigersinn, welche der ursprünglichen Rotation entgegenwirkt. Anschließend müssen die x- und y-Werte der vorhergesagten Differenzen nur noch passend zusammengesetzt und aufaddiert werden, um die finalen vorhergesagten Touchpunkte für die einzelnen Vorhersagestufen zu erhalten. Die Vorhersageroutine übernimmt außerdem die Skalierung der IMU-Werte (beschrieben in Kapitel 5.1), welche vom IMU-Handschuh in Rohform geliefert werden, bevor diese als Eingaben verwendet werden. Die Routine kann dann

[4]Erläuterungen zu TensorFlow-Werkzeugen: https://www.tensorflow.org/extend/tool_developers/
[5]TensorFlow Mobile: https://www.tensorflow.org/mobile/
[6]JNI-Entwicklerrichtlinie: https://developer.android.com/training/articles/perf-jni.html

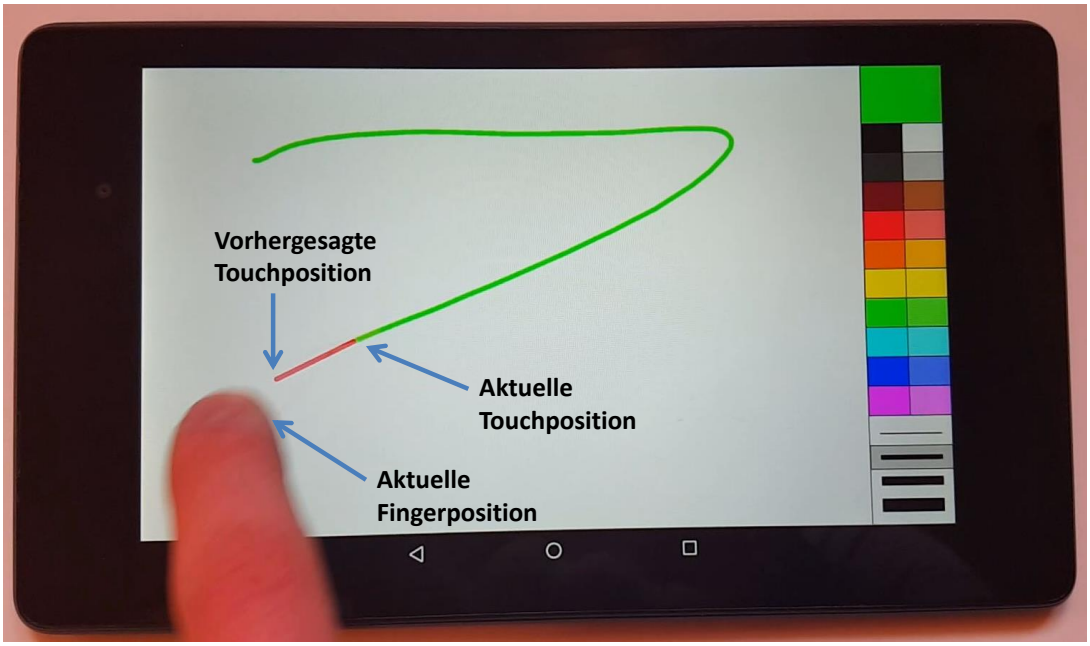


Abbildung 5.6: Zu sehen ist das Bild einer aktiven 66ms Vorhersage des reinen Touchmodells für Fingereingaben. Die rot eingefärbte Linie (Einfärbung nur zur besseren Darstellung) stellt die vorhergesagte Trajektorie in 66ms Zukunft dar. Sie schließt die Lücke zwischen realer Fingerposition und zuletzt erkannter Touchposition zu einem beträchtlichen Teil.

im Android-Code aufgerufen werden, mit den letzten elf Touchpunkten und den letzten zehn IMU-Paketen als Parameter, welche sie dann verarbeitet, an das gewählte Modell übergibt und anschließend die Vorhersagen zurückliefert.

Die Vorhersagen sollen aktiv zu jedem Zeitpunkt auf dem Tablet genutzt werden können, um die Kompensierung von Latenzen jederzeit zu ermöglichen, wie etwa auf Abbildung 5.6, für die Zeichenapplikation aus der Datenerhebungsstudie, zu sehen. Dabei gibt es einen kritischen Faktor zu berücksichtigen, nämlich die aktiven Kosten einer Vorhersage auf dem Tablet. Dafür wurden für die vier ausgewählten und implementierten Modelle die Durchschnittszeiten gemessen (je über 1000 Vorhersagen), welche eine Vorhersage benötigt, während aktiver Benutzung in der Zeichenapplikation, wie auf dem Bild 5.6. Für das reine Touchmodell für Fingereingaben ergab sich so ein Durchschnitt von 2,60ms (SD = 0,86ms) und für das reine Touchmodell mit Styluseingaben analog ein Durchschnitt von 2,49ms (SD = 0,79ms). Die entsprechenden Kosten für die IMU-Modelle sind durchschnittlich 3,98ms (SD = 0,83ms) für das $IMU_{Handgelenk}$-Modell für Fingereingaben und 3,94ms (SD = 0,94ms) für das IMU_{Stylus}-Modell für Styluseingaben.

Die Kosten für die IMU-Modelle sind demnach um 34,7% für Fingereingaben und 36,6% für Styluseingaben höher, als die der reinen Touchmodelle, jedoch sind die Auswirkungen gering. Aus einer 66,6ms Vorhersage wird bei kontinuierlicher Vorhersage mit einem IMU-Modell etwa eine 62,6ms Vorhersage und aus einer 33,3ms Vorhersage eine 29,3ms Vorhersage, was laut Deber et al. noch deutlich zu spüren sein sollte [Deb+15]. Diese Form der kontinuierlichen, aktiven Vorhersage eignet sich also für die aktive Nutzung und kann demnach in einer zweiten Studie, im Hinblick auf Performanzverbesserungen der Nutzer, evaluiert werden.

5.7 Zusammenfassung

In diesem Kapitel wurden die gesammelten Daten aus dem letzten Kapitel dazu verwendet, zehn unterschiedliche neuronale Netze für Touchvorhersagen zu trainieren. Alle der resultierenden Modelle, welche IMUs für die Vorhersagen verwendeten, konnten eine höhere Genauigkeit ihrer Vorhersagen als die reiner touchbasierter Modelle auf den gleichen Daten vorweisen. Für 33ms Vorhersagen konnte die durchschnittliche Genauigkeit von Vorhersagen für Finger- und Styluseingaben um über 9% gesteigert werden, für 66ms Vorhersagen sogar um über 21% und für 99ms Vorhersagen um über 29% für Fingereingaben und über 30% für Styluseingaben (Modell: Touch und IMU_{Finger} und IMU_{Stylus}). Zwei durchgeführte ANOVAs bestätigten, dass die verbessernden Effekte durch die IMUs signifikant sind. Um den Einfluss verschiedener Faktoren, der Vorhersagen einzelner Modelle, auf die Performanz von Anwender zu untersuchen, wurden vier der trainierten Modelle ausgewählt, auch im Hinblick auf die zeitnahen Integrationsoptionen der Modelle in kommerzieller Hardware. Diese unterscheiden sich in den zu untersuchenden Faktoren (Vorhersage, Vorhersagestufe, Eingabemethode und IMU-Nutzung), was einen Vergleich der Einflüsse dieser Faktoren in der folgenden Evaluationsstudie zulässt. Zuletzt wurden die Modelle in die Android-Applikation eingebunden und eine Routine erstellt, welche kontinuierliche, aktive Vorhersagen zu jedem Zeitpunkt mit geringen Kosten, von durchschnittlich unter 4ms, erlaubt.

6 Evaluation der Modelle

Bisher wurde gezeigt, dass IMUs die Genauigkeit von Vorhersagemodellen zur Latenzverringerung verbessern. Ob diese Verbesserungen aber von Nutzern überhaupt wahrgenommen werden, lässt sich dabei nicht feststellen. Aus diesem Grund wurde im Verlauf der Arbeit eine zweite Studie durchgeführt, mit welcher die Effekte der vier gewählten Vorhersagemodelle (Für Fingereingaben das reine Touchmodell und das Touch- und $IMU_{Handgelenk}$-Modell und für Styluseingaben das reine Touchmodell basierend auf Stylusdaten und das Touch- und IMU_{Stylus}-Modell) evaluiert werden sollten. Vor allem die Effekte auf die Performanz der Nutzer und die Benutzererfahrung sollten dabei untersucht werden. Mit diesen beiden Eigenschaften lässt sich sowohl ein objektiver, als auch ein subjektiver Vergleich durchführen. Demnach sollte die Performanz objektiv und die Benutzererfahrung subjektiv gemessen werden. Die Hypothese für die objektive Messung ist, dass Nutzer mit höheren Vorhersagestufen performanter sind, als mit niedrigeren und mit den IMU-Modellen, aufgrund der höheren Genauigkeit, ebenfalls performanter als mit den IMU-losen Modellen. Für das subjektive Feedback und den Unterschied zwischen Finger- und Styluseingaben wurden keine Hypothesen aufgestellt. Die unabhängigen Variablen für diese Studie waren die Eingabemethoden (Finger oder Stylus), die Nutzung von IMUs (Ja oder Nein) und die Vorhersagestufen (0ms (Ausgangsbasis), 33ms und 66ms). Als nächstes wird das gewählte Studiendesign erläutert, bevor auf Studiendetails eingegangen wird.

6.1 Design

Da die Nutzung von IMUs nur für 33ms und 66ms Vorhersagestufen getestet werden kann (bei 0ms erfüllt die IMU keinen Zweck und verändert somit auch keine Ergebnisse), wird die unabhängige Variable der Vorhersagestufen in den Faktor der IMU-Nutzung eingenistet, also hierarchisch untergeordnet. Man spricht bei einem solchen Studiendesign von einem „Nested Design“ und es ermöglicht, die Varianz in der Hierarchie der unteren Messdaten zu untersuchen [KAB14]. Die Struktur des in dieser Arbeit verwendeten „Crossed and Nested“-Designs ist in Abbildung 6.1 dargestellt. Das bedeutet, dass nur ein Teil des Designs, beziehungsweise nicht alle unabhängigen Variablen, eingenistet sind. In diesem Fall sind das nur die Vorhersagestufen, die restliche Variablen sind weiterhin kreuzfaktorisiert (auch: kreuzverbunden, englisch: „crossed“), deshalb der Name des Designs.

Das bedeutet, in Bezug auf die hier beschriebene Studie, dass am Ende die Effekte zwischen IMU-Nutzung und keiner IMU-Nutzung zwar verglichen werden können, auch in Bezug auf Stylus

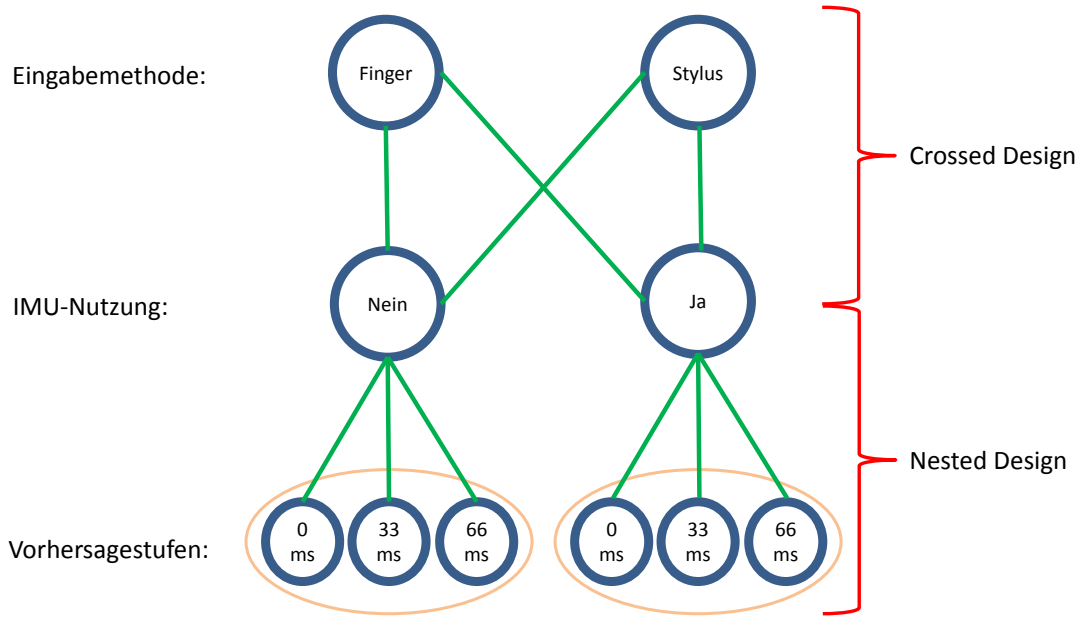


Abbildung 6.1: Hier wurde das experimentelle Design der Studie mit allen unabhängigen Variablen und möglichen Belegungen als Plandiagramm dargestellt. Es wurde markiert, welche Teile des Design Crossed (kreuzfaktorisiert) und welche Teile Nested (eingenistet) sind. Die grünen Verbindungslinien entsprechen den zehn Konditionen des Designs.

oder Finger, aber nicht zwischen den Vorhersagestufen der unterschiedlichen Konfigurationen verglichen werden kann. Innerhalb eines Unterknotens für die IMU-Nutzung können die Effekte zwischen den einzelnen Vorhersagestufen 0ms, 66ms und 99ms weiterhin untersucht werden, es können lediglich keine Vergleiche zwischen beispielsweise der 33ms Vorhersage eines Modells mit IMU und der 33ms Vorhersage eines Modells ohne IMU erforscht werden. Anders ausgedrückt bedeutet das, dass keine Kreuzvergleiche mehr zwischen den unabhängigen Variablen Nutzung von IMUs (Ja oder Nein) und Vorhersagestufen (0ms, 33ms und 66ms) und Eingabemethoden (Finger oder Stylus) und Vorhersagestufen (0ms, 33ms und 66ms) mehr möglich sind. Dafür erlaubt dieses Design die Kombination von IMU-Nutzung und 0ms Vorhersage, welche eigentlich nicht existiert, aber in dieser Studie gemessen wurde. Würde es das nicht oder würde man ein anderes Design zur Auswertung verwenden, würden die nicht existierenden Modelle, beziehungsweise die doppelt ausgeführten Modelle mit 0ms Vorhersage, die statistische Relevanz der anderen Konditionen reduzieren, da signifikante

Unterschiede vereinheitlicht werden würden. Kreuzvergleiche zwischen der Eingabemethode und der IMU-Nutzung sind weiterhin möglich, wie auch in Abbildung 6.1 zu sehen.

Bei der hier vorgestellten Studie handelt es sich insgesamt um ein experimentelles, „repeated measures"- (entspricht within-subjects Design; jeder Teilnehmer führt alle Aufgaben unter allen Bedingungen durch) „Crossed and Nested"-Design mit zehn verschiedenen Konditionen. Diese können wiederum in Abbildung 6.1 betrachtet werden, da jede Kante dort einer Kondition entspricht (zum Beispiel die Kante zwischen Eingabemethode (Finger) und IMU-Nutzung (Ja) repräsentiert die Kondition, welche die Effekte zwischen diesen beiden Variablenwerten untersucht). Später sollen alle Konditionen über eine ANOVA ausgewertet werden.

Nach der Definition des Studiendesigns soll nun weiter auf die Studie an sich eingegangen werden. Im Folgenden wird die Durchführung der Studie im Detail beschrieben und anschließend werden die Ergebnisse präsentiert. Über eine ANOVA wird danach festgestellt, ob die Ergebnisse signifikante Unterschiede aufweisen, im Hinblick auf die Konditionen des verwendeten Studiendesigns.

6.2 Evaluationsmethode

Bei der zweiten Studie handelt es sich, genau wie bei der ersten, um ein experimentelles Labordesign. Die kontrollierten Variablen sind dabei das verwendeten System, welches aus der ersten Studie übernommen wurde, und die Position des Tablets, welches bei allen Tests flach auf dem Tisch liegen musste, weil die neuronalen Netze fast ausschließlich auf solchen Daten trainiert wurden. Weiterhin durften nur Rechtshänder an der Studie teilnehmen und sie wurde in einer laborähnlichen Umgebung durchgeführt, mit vorgeschriebenen Aufgaben. An dem System aus der ersten Studie, mit den bereits integrierten Änderungen aus dem letzten Kapitel, mussten für die Evaluierung noch einige Modifikationen vorgenommen werden, vor allem zur Visualisierung der Vorhersagen. Diese werden nun für die einzelnen Aufgaben kurz angeführt. Darauf folgt die Beschreibung der Prozedur der Studie zur Evaluierung.

Modifikationen des Systems

An dem externen Sensorprototyp, beziehungsweise dem Handschuh, wurden keine Änderungen seit der ersten Studie durchgeführt. Die Android-Applikation wurde leicht verändert, um die Vorhersagen in der Applikation zu nutzen, wie im letzten Kapitel beschrieben. Die vier ausgewählten Modelle, der trainierten neuronalen Netze, wurden eingebunden und einzeln aktivierbar gemacht. Die Aktivierung konnte anschließend über ein Menü vorgenommen werden, in welchem sich sowohl das Modell, als auch die Vorhersagestufe wählen ließen. Ist ein Modell aktiviert, so werden kontinuierlich Vorhersagen durchgeführt. Zur Darstellung der vorhergesagten Touchpunkte wurden beim Fitts' Law-Test und bei der Zeichenanwendung die Zeichenmethoden abgeändert. Im Fitts' Law-Test wird das kleinere, zu manipulierende

Kästchen immer an der Position der Vorhersage, der gewählten Vorhersagestufe (33ms oder 66ms), gezeichnet, statt an der zuletzt vom Touchscreen erkannten Touchposition, wie üblich für Touchgeräte. Bei der Zeichenumgebung der bisher verwendeten Applikation wird die Vorhersage als Trajektorie aller vorhergesagten Punkte, bis zur gewählten Vorhersagestufe, an den zuletzt vom Touchscreen erkannten Touchpunkt angehängt. Für eine 66ms Vorhersage werden demnach die vorhergesagten Touchpositionen in 16,6ms, 33,3ms, 49,9ms und 66,6ms Entfernung gezeichnet, über Linien zu einer Trajektorie verbunden und über den vorhergesagten 16,6ms Punkt zum letzten erkannten Touchpunkt verbunden. Die vorhergesagte Trajektorie wird mit der gleichen Strichstärke und Farbe wie die originale Trajektorie eingezeichnet, besteht aber nur bis zur nächsten Vorhersage, von welcher sie überschrieben wird. So wird vermieden, dass fehlerhafte Vorhersagen bestehen bleiben und auf diese Weise das entstehende Bild verändern. Trotzdem schließt die temporäre Vorhersage visuell die Lücke, welche durch Latenzen erzeugt wird, wie in Abbildung 5.6 zu sehen war. Die Schreibumgebung wurde auf dieselbe Weise modifiziert, wie die Zeichenumgebung. Das Google Nexus 7 Tablet und die genutzte Android-Applikation wurden, bis auf die gerade beschriebenen Änderungen zur Nutzung von Vorhersagen, nicht verändert. Lediglich die Protokollierungs- und Speicherfunktionen für die Touch- und Sensordaten wurden, im Gegensatz zur ersten Studie, deaktiviert, um stabilere Performanzraten des Tablets gewährleisten zu können.

Prozedur

Auch für diese Studie wurde ein „within-subjects“-Design verwendet (beziehungsweise „repeated measures“), was bedeutet, dass jeder Teilnehmer alle Aufgaben unter allen Bedingungen durchführen musste. Im Verlauf der Studie musste jeder Teilnehmer zwei Aufgaben je zwölfmal durchführen, welche weiter unten beschrieben werden. Die Reihenfolge der einzelnen Aufgaben für jeden Teilnehmer war randomisiert, mit der Bedingung, dass jede randomisierte Reihenfolge nur einmal vorkommen darf. Durch die Bedingung werden die Reihenfolgen also pseudo-randomisiert. Die pseudo-randomisierten Reihenfolgen wurden im Vorfeld definiert. Dadurch wurden Sequenzeffekte vermieden. Die zwölf Durchläufe pro Aufgabe ergeben sich aus den unabhängigen Variablen. Jede Aufgabe musste mit jedem der vier gewählten Modelle je mit den Vorhersagestufen 0ms, 33ms und 66ms durchgeführt werden, was in zwölf Durchläufen pro Aufgabe resultiert.

Die Studie dauerte pro Teilnehmer etwas über eine Stunde und hatte denselben Ablauf, wie die erste Studie: Jeder Teilnehmer wurde nach dem Unterzeichnen einer Einverständniserklärung gebeten, den Handschuh mit den Inertialsensoren anzulegen und erhielt eine kurze Unterweisung. Dabei wurde erklärt, dass es sich um eine Studie zur Evaluation von Latenzkompensierungsmodellen handelt und wie die Aufgaben aufgebaut sind. Anschließend bekam jeder Teilnehmer eine numerische, anonymisierte ID zur späteren Zuordnung der Daten und eine feste, vordefinierte, pseudo-randomisierte Reihenfolge der Aufgaben. Nach der Auswahl des ersten Vorhersagemodells mit der jeweilige Vorhersagestufe und dem Start der Aufgaben durch den Betreuer wurde die Zeit gestoppt und die Teilnehmer durften beginnen. Zwischen

jeder Aufgabe wurde vom Betreuer der Studie überprüft, ob ein Fehlverhalten der externen Inertialsensoren feststellbar ist, etwa durch das Lockern von Verbindungen durch rapide Bewegungen. Anschließend wurde das nächste Modell mit der entsprechenden Vorhersagestufe vom Betreuer geladen. Die Auswahl der Modelle und Stufen geschah immer ohne die Einsicht der Teilnehmer, damit ihre Ergebnisse nicht beeinflusst wurden. Während den Aufgaben wurden Hardware-Tasten am Gerät über die Android-Applikation deaktiviert, um Fehler zu vermeiden und der Flugmodus des Android-Betriebssystems wurde aktiviert, um Performanzeinbrüche zu reduzieren. Die Aufgaben wurden alle im Sitzen an einem Tisch mit angelegtem Sensorhandschuh durchgeführt.

Nach dem Absolvieren aller Aufgaben wurden den Teilnehmern in einem anschließenden semi-strukturierten Interview noch bis zu fünf Fragen in fester Reihenfolge und Abhängigkeit gestellt. Auf diese sollten die Teilnehmer frei antworten. Das qualitative Feedback wurde gesammelt, um später die Relevanz dieser Arbeit zu verdeutlichen. Die gewählten Fragen lauteten wie folgt:

1.) Hast du jemals auf einem deiner Touchgeräte Latenzen/Lag/Verzögerungen bei Touchbewegungen verspürt?

   2.) Falls ja, bei welchen Tätigkeiten?

   3.) Falls ja, hat dich das gestört?

4.) Kannst du dir Touch-Aufgaben vorstellen, bei welchen eine reduzierte/kleinere Latenz Vorteile bringt?

   5.) Falls ja, welche Vorteile (bei welcher Aufgabe) wären das?

Zur objektiven Messung von Performanzunterschieden und zur subjektiven Messung von Benutzererfahrungen wurden die folgenden Aufgaben ausgewählt, welche bereits durch die Visualisierung von Vorhersagen erweitert wurden.

6.2.1 Aufgaben

Die Aufgabentypen aus der ersten Studie wurden nicht verändert, lediglich auf zwei Aufgaben beschränkt. Dabei handelt es sich um die Fitts' Law-Aufgabe und die Zeichenaufgabe. Die Schreibaufgabe wurde aus verschiedenen Gründen vernachlässigt. Zum einen lassen sich beim Schreiben eher subjektive als objektive Ergebnisse sammeln, was bereits bei der Zeichenaufgabe geschieht. Zum anderen war die Genauigkeit der hier erstellten Modelle bei Schreibaufgaben relativ gering, aufgrund von einem weniger umfangreichen Datensatz von Schreibdaten, wie etwa bei Henze et al., bei denen die Touchtrajektorien für Schreibaufgaben besser erlernt werden konnten [HFS16]. Zur Hervorhebung der Stärken der trainierten Modelle wurde demnach die Zeichenaufgabe gegenüber der Schreibaufgabe bevorzugt. Die Schreibaufgabe wurde folglich weggelassen, um den Rahmen der Studie nicht zu sprengen.

Die Fitts' Law-Aufgabe wurde nicht verändert, lediglich, wie oben beschrieben, mit den Vorhersagen ergänzt. Weiterhin wurde bei Fitts' Law-Aufgaben die benötigte Zeit, Aufgabendetails (Entfernung, Zielgröße, Zielposition) und Genauigkeit (Ziel getroffen oder nicht) für jede einzelne der 72 Aufgaben eines Durchlaufs gemessen und gespeichert, wie in der ersten Studie, in welcher diese Daten zwar erhoben, aber nicht verwendet wurden. Ein Unterschied zur ersten Studie ist lediglich, dass der Name des verwendeten Vorhersagemodells und die Stufe der Vorhersage bei der Dateierstellung der erhobenen Daten mitgespeichert wurden, damit die Daten später den einzelnen Modellen und Vorhersagestufen zugeordnet werden konnten. In dieser Studie dienten diese Daten dann zur objektiven Messung von Nutzerperformanz durch die Berechnung des Fitts' Law-Durchsatzes nach MacKenzie [Mac92]. Dafür werden weiterhin die Formeln 6.1 und 6.2 verwendet.

$$ID = \log_2(\frac{D}{W} + 1) \tag{6.1}$$

$$IP = \frac{ID}{MT} \tag{6.2}$$

Die Gleichung für den „index of performance" (IP) in Formel 6.2 über die Verwendung des „index of difficulty" (ID), in Formel 6.1, und der benötigten Zeit (MT) ist dabei der Schlüssel für eine objektive Messung. Dieser misst den Durchsatz der Anwender in Bits pro Sekunde, abhängig von der Schwierigkeit der Aufgabe, in Bezug auf Distanz zum Ziel (D) und Breite des Ziels (W), welcher hier für die Messung der Performanz genutzt wird [Fit54; Mac92]. Den Fitts' Law-Test mussten die Teilnehmer jeweils zwölfmal ausführen, für alle Kombinationen der vier Modelle und drei Vorhersagestufen, wobei sie wieder so schnell und genau wie möglich sein sollten. Lag die Fehlerrate der Nutzer unter 5%, wurden sie animiert, schneller zu verfahren.

Die Zeichenaufgabe wurde bis auf die temporäre Zeichnung der gewählten Vorhersagen auch nicht variiert. Sie wurde auch zwölfmal pro Teilnehmer für alle Modell- und Stufenkombinationen durchgeführt, mit einem Zeitlimit von zwei Minuten pro Durchführung. Die Zeichnungsvorgabe bestand diesmal aus zwölf Fotos von tropischen Urlaubsinseln und Stränden. Dies sollte gewährleisten, dass alle Teilnehmer ähnliche Formen und Muster malen, um eine gleiche Basis für die Benutzererfahrungen zu schaffen. Würden beispielsweise zwei Teilnehmer komplett verschiedene Formen zeichnen, etwa einer nur gerade Linien und der andere nur bogenförmige, so könnte sich die Benutzererfahrung massiv unterscheiden, da bestimmte Modelle eventuell bogenförmige Bewegungen sehr schlecht vorhersagen, geradlinige aber dafür sehr gut. Die Reihenfolge der Modelle und Vorhersagezeiten war pseudo-randomisiert, die Reihenfolge der Vorlagenbilder aber fest. So wurde gewährleistet, dass ein bestimmtes Modell nicht aufgrund eines ungeeigneten Vorlagenbildes, bei jedem Durchgang jedes Nutzers, schlechte Benutzererfahrungen erzeugt. In Abbildung 6.2 sind einige Beispiele von gemalten Teilnehmerbildern, nach den Vorlagenfotos aus der Studie, zu finden.

Zur Messung der subjektiven Benutzererfahrung und der Benutzbarkeit des Systems wurden die Teilnehmer gebeten, nach jedem der zwölf Zeichen- oder Fitts' Law-Aufgabendurchläufe einen Bogen von drei Fragen zu beantworten. Darin wurde gefragt, ob sie Jitter/Zittern bei den

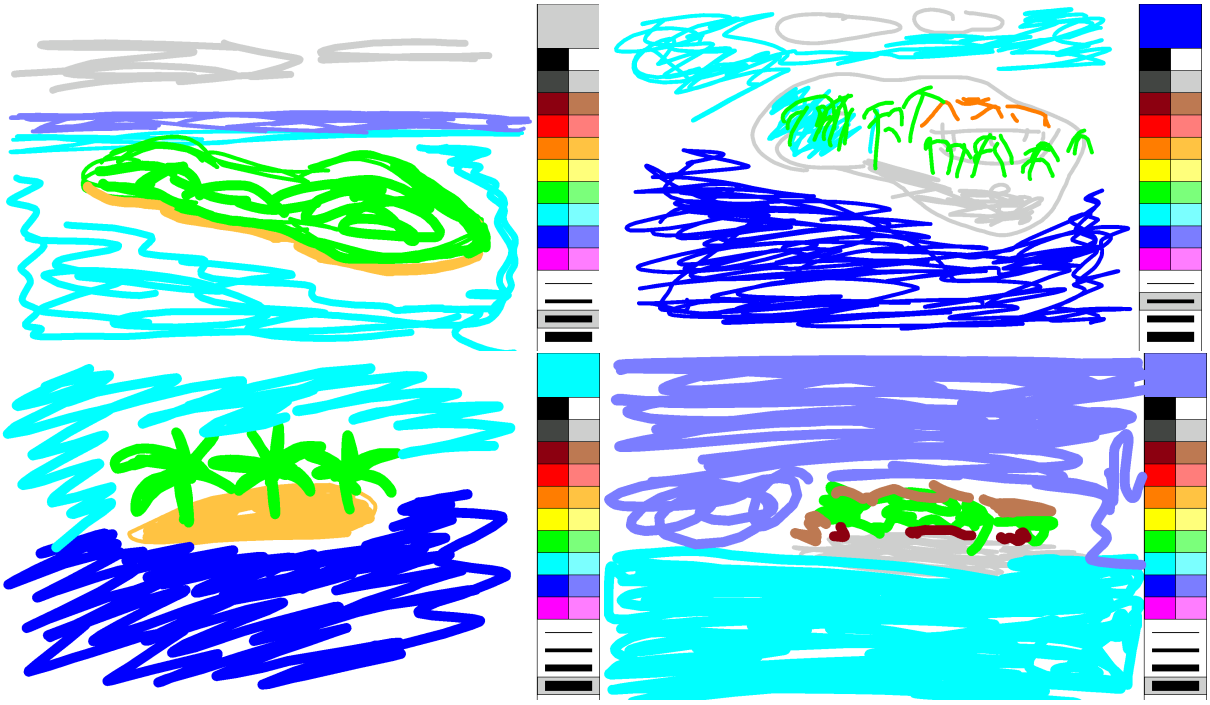


Abbildung 6.2: Die Abbildungen stellen vier Zeichnungen dar, welche während der zweiten Studie von den Teilnehmern, nach den Vorlagen, gemalt wurden.

Toucheingaben verspürt haben, ob sie Latenzen/Verzögerungen verspürt haben und ob sie es als unerfreulich empfanden, den Touchscreen (mit der entsprechenden Eingabe) zu verwenden. Jede Frage musste durch Ankreuzen von eines der sieben Kästchen beantwortet werden. Das erste Kästchen stand dabei für „Ich stimmte nicht zu“, also der besten Bewertung für eine Eingabemethode und das siebte Kästchen für „Ich stimmte zu“, demnach der schlechtesten Bewertung für eine Eingabemethode. Der genaue englische Wortlaut der Fragebögen ist in Abbildung 6.3 dargestellt. Es handelt sich hierbei um eine sieben Punkte Likert-Skala [Lik32].

6.2.2 Teilnehmer

An der Studie nahmen 16 Teilnehmer teil. Acht davon waren weiblich und acht männlich. Je 50% der männlichen und weiblichen Teilnehmer hatten zuvor an der ersten Studie teilgenommen. Acht Teilnehmer wurden zufällig am Campus der Universität Stuttgart ausgewählt, die restlichen acht nach Terminvereinbarung in Böblingen. Sechs Teilnehmer waren Studenten verschiedener technischer Studiengänge in verschiedenen Semestern. Die restlichen zehn Teilnehmer waren in einem festen Arbeitsverhältnis eines nicht technischen Bereichs. Das Alter der Teilnehmer reichte von 20 Jahren bis 61 Jahren, mit einem Durchschnitt von 28,81 Jahren (SD = 11,9 Jahre). Alle Teilnehmer waren Rechtshänder und hatten keine motorischen Störungen angegeben. Die Handgrößen, gemessen vom Handgelenk zur Mittelfingerspitze, reichten von 16,1cm bis 20,1cm bei einem Durchschnitt von 17,98cm (SD = 1,2cm). Weiterhin

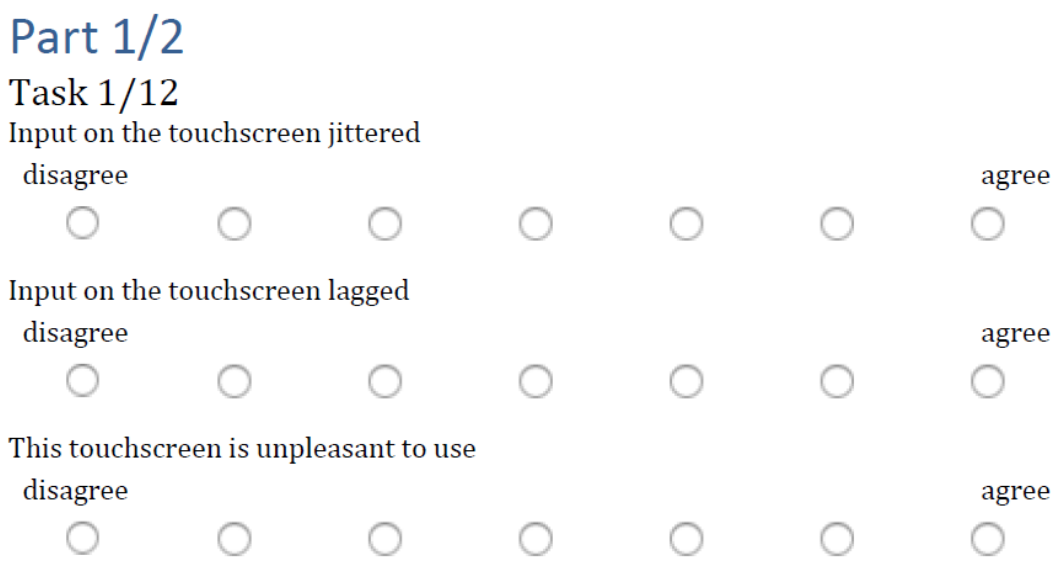


Abbildung 6.3: Zu sehen ist ein Auszug aus einem Fragebogen eines Teilnehmers der zweiten Studie, welche diese zu jedem der 24 Aufgabendurchläufe ausfüllen mussten. Die Fragebögen waren ausschließlich auf Englisch verfügbar. Die drei Fragen wurden immer als Block mit einer Zahl versehen, um sie später dem Modell zuzuordnen, zu welchem sie beantwortet wurden, ohne dass die Teilnehmer die Modellbezeichnung erfahren. Das Beantworten wurde durch Ankreuzen eines der Kästchen, durch den Teilnehmer, mit einem Kugelschreiber, vollzogen.

wurde von allen Teilnehmern die Breite und Länge des verwendeten Zeigefingers protokolliert, was eine durchschnittliche Breite von 1,9cm und eine Länge von 7,9cm offenbarte.

Alle Teilnehmer haben im Zeitraum der Studie täglich einen Touchscreen verwendet. Nur fünf Teilnehmer verwendeten regelmäßig einen Touchstylus. Von den fünf verwendete nur eine Person täglich einen Stylus, drei Personen wöchentlich und die letzte monatlich. Die restlichen Teilnehmer gaben an, nie einen Stylus für Touchgeräte zu verwenden. Fast alle Teilnehmer, bis auf jeweils eine Person, nutzten ihre eigenen Touchgeräte regelmäßig für das Surfen im Internet und zum Chatten. Dreizehn Teilnehmer schrieben weiterhin regelmäßig E-Mails, zwei spielten und vier editierten Texte. Vor der Studie wurden den Teilnehmern erklärt, dass sie jederzeit Pausen einlegen können. Niemand der Teilnehmer legte eine Pause während der Studie ein. Sie wurden ebenfalls darauf hingewiesen, das Tablet auf dem Tisch liegen zu lassen, während der Durchführung von Aufgaben. Am Ende wurden alle Teilnehmer mit 10€ für ihre Teilnahme entlohnt. Bei keinem Teilnehmer kam es zu bemerkbaren Sensorausfällen.

6.3 Ergebnisse

Die Ergebnisse gliedern sich in objektive Ergebnisse, des Fitts' Law-Tests, und subjektive Ergebnisse. Zu den subjektiven Ergebnissen gehören hier die Fragebögen und die Interview-Antworten. Die Ergebnisse werden hier lediglich vorgestellt, die Diskussion findet erst im Anschluss statt. Auf allen Ergebnissen wurden ANOVAs durchgeführt, welche im Zusammenhang mit den jeweiligen Ergebnissen beschrieben werden.

6.3.1 Ergebnisse der Fitts' Law-Aufgabe

Insgesamt wurden 13824 Fitts' Law-Aufgaben aufgezeichnet. Das ergibt sich aus den zwölf Durchläufen mit je 72 Wiederholungen jedes der 16 Teilnehmer. Von diesen Daten wurden lediglich die Daten erfolgreicher Aufgaben verwendet, bei denen das Ziel getroffen wurde, was in 13424 übrigen Daten resultierte. Davon wurden wiederum die untersten und obersten 1% als Ausreißer deklariert und herausgefiltert. Die 0ms Vorhersage Kondition wurde für beide Eingabemethoden im Verlauf der Studie von jedem Teilnehmer zweimal durchgeführt. Die Daten der gleichen Konditionen wurden jeweils zusammengefasst, um auf die zehn angestrebten Konditionen zu kommen. Die Ergebnisse der 0ms Konditionen basieren demnach auf der doppelten Datengrundlage, wie die der anderen Konditionen. Die zwei 0ms Vorhersage Konditionen bilden die Ausgangsbasis für Toucheingaben ohne jegliche Verbesserung.

Mit den übrigen Daten wurden zuerst die durchschnittlichen Zeiten zur Bewältigung von einzelnen Fitts' Law-Aufgaben berechnet, welche in Abbildung 6.4 dargestellt sind. Zu sehen ist hierbei, dass die geringsten Durchschnittswerte von den 66ms Vorhersagen der Modelle mit IMU-Nutzung erzeugt wurden. Die Teilnehmer waren mit den IMU-Modellen durchschnittlich um 51,0ms schneller bei Fingereingaben und 29,0ms bei Styluseingaben, gegenüber den IMU-losen Modellen mit 66ms Vorhersagen. Das entspricht einer durchschnittlichen Verbesserung von 7,7% für Fingereingaben und 4,5% für Stylusaufgaben bei 66ms Vorhersagen. Bei den 33ms Vorhersagen fällt diese Verbesserung durch IMUs sehr viel geringer aus (0,2% Verschlechterung für Fingermodelle und 1,1% Verbesserung für Stylusmodelle). Weiterhin lässt sich beobachten, dass die Konditionen ohne Vorhersage die höchsten Durchschnittszeiten, für den jeweiligen Eingabetyp, bei den Teilnehmern erzeugen. Es lässt sich ebenfalls eine Tendenz erkennen, dass mit steigender Vorhersagestufe die benötigte Aufgabenzeit sinkt. Dies trifft aber nicht auf das reine Touchmodell für Fingereingaben zu, bei welchem die durchschnittliche Aufgabenzeit der Teilnehmer bei der 66ms Vorhersage höher ist, als bei der 33ms Vorhersage. Zur Betrachtung der Verbesserungen im Detail reicht die Durchschnittszeit alleine allerdings nicht aus.

Weiterhin lässt sich die Genauigkeit der Teilnehmer durch die Anzahl getroffener Ziele im Verhältnis zur Gesamtzahl der Ziele bei den einzelnen Modellen und Konditionen berechnen. Die durchschnittliche Genauigkeit der Teilnehmer der zweiten Studie wurde für jede Kondition in Abbildung 6.5 dargestellt. Die größten Abweichungen lassen sich wieder bei den 66ms der reinen Touchmodelle erkennen. Sie weisen für beide Eingabemethoden den geringsten

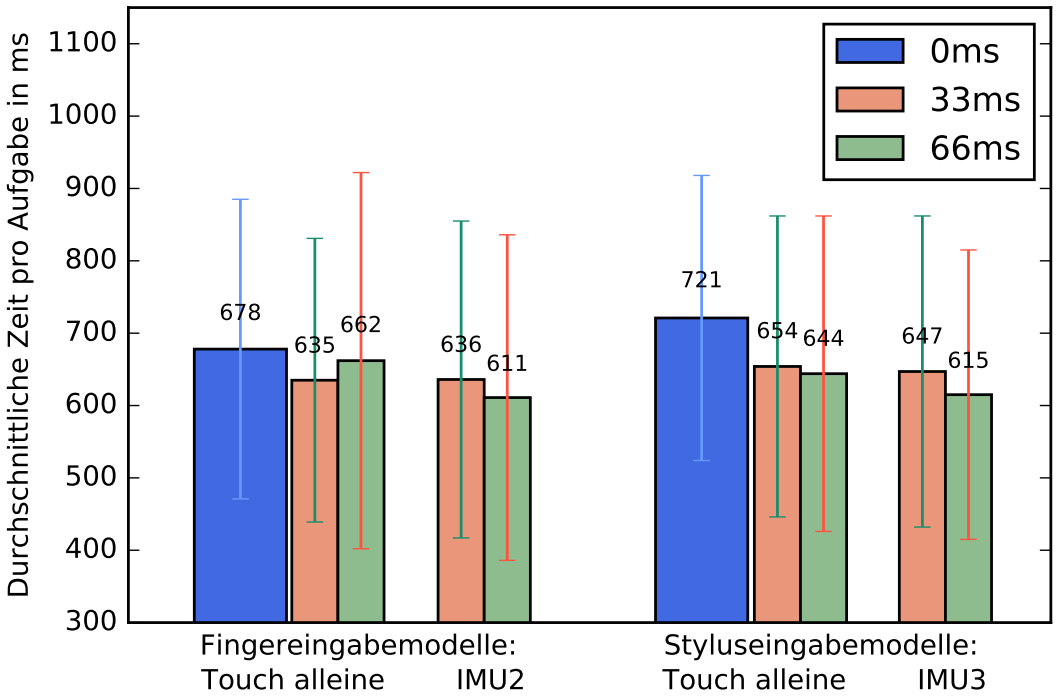


Abbildung 6.4: Die Abbildung stellt die durchschnittliche Zeit zur Bewältigung einer einzelnen Fitts' Law-Aufgabe als Balkendiagramm in Millisekunden dar. Die Balken wurden räumlich nach ihren vier zugehörigen Modellen angeordnet, welche darunter angeführt werden. Die Abkürzung IMU2 entspricht dem Modell Touch- und $\text{IMU}_{\text{Handgelenk}}$ für Fingereingaben und IMU3 entspricht dem Modell Touch- und $\text{IMU}_{\text{Stylus}}$ für Stylusaufgaben. Die 0ms Modelle wurden mit doppelter Breite dargestellt, aufgrund der doppelten Datengrundlage. Die Durchschnittswerte wurden an den Balken annotiert. Die Vorhersagestufen wurden farbkodiert. Die Fehlerbalken zeigen die Standardabweichung.

Genauigkeitswert auf. Alle anderen Unterschiede sind marginal, da sie sich fast alle anderen Genauigkeitswerte in einem Bereich von 97% bis 98% befinden.

Der Durchsatz der Teilnehmer gibt weitaus mehr Aufschluss über Verbesserungen zwischen Konditionen. In Abbildung 6.6 wurde deshalb der durchschnittliche Durchsatz der Studienteilnehmer bei den einzelnen Konditionen dargestellt. Dieser wurde nach dem bereits beschriebenen „index of performance" nach MacKenzie für Fitts' Law-Aufgaben in $Bits/s$ berechnet [Mac92]. Zu erkennen ist, dass die Ausgangslage, die 0ms Vorhersagen ohne Verbesserungen, bei den Teilnehmern die geringsten durchschnittlichen Durchsätze für jede Eingabemethode ermöglichte. Wie erwartet führt jede andere Kondition zu Verbesserungen. Diese fallen unterschiedliche stark aus. Die größte Verbesserung ist bei dem Modell Touch- und $\text{IMU}_{\text{Handgelenk}}$ für Fingereingaben zu erkennen, welches gegenüber der Ausgangsbasis von 0ms durchschnittliche Durchsatzverbesserungen von 15,5% bei 66ms Vorhersagen ermöglichte. Das

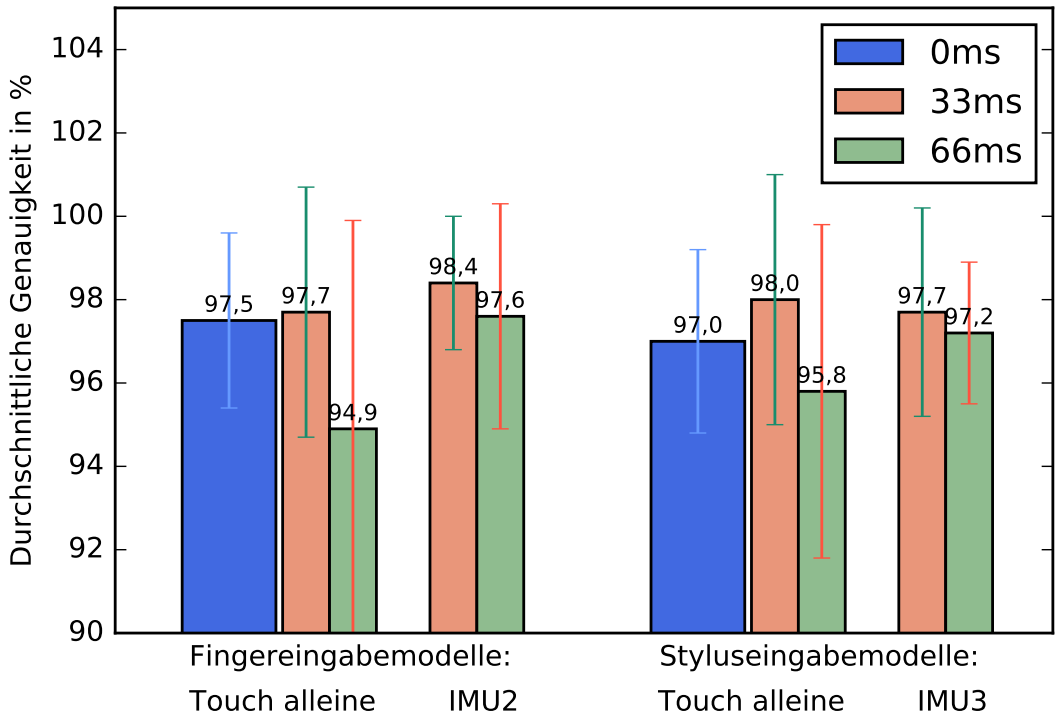


Abbildung 6.5: Die Abbildung stellt die durchschnittliche Genauigkeit bei der Bewältigung des Fitts' Law-Tests als Balkendiagramm in % dar. Die Balken wurden räumlich nach ihren vier zugehörigen Modellen angeordnet, welche darunter angeführt werden. Die Abkürzung IMU2 entspricht dem Modell Touch- und $IMU_{Handgelenk}$ für Fingereingaben und IMU3 entspricht dem Modell Touch- und IMU_{Stylus} für Stylusaufgaben. Die 0ms Konditionen wurden mit doppelter Breite dargestellt, aufgrund der doppelten Datengrundlage. Die Durchschnittswerte wurden an den Balken annotiert. Die Vorhersagestufen wurden farbkodiert. Die Fehlerbalken zeigen die Standardabweichung.

entsprechende IMU-Modell für Styluseingaben ermöglicht gegenüber seiner Ausgangsbasis sogar Durchsatzverbesserungen von 17,1%, wenn eine 66ms Vorhersage genutzt wurde. Die Verbesserungen bei 33ms Vorhersagen fallen etwas geringer aus mit 9,5% für das IMU-Modell mit Fingereingaben und 13,5% für das mit Styluseingaben. Auch die IMU-losen Modelle weisen Verbesserungen gegenüber der Ausgangsbasis auf, jedoch in geringerem Maße als die IMU-Modelle. Die IMU-Modelle weisen um 2,2% bis 8,2% bessere durchschnittlichen Durchsätze auf, als die der entsprechenden IMU-losen Modelle. Bei den Fingereingaben ist das IMU-Modell bei 33ms Vorhersagen um 3,6% und bei 66ms Vorhersagen um 8,2% besser, als das IMU-lose Gegenstück. Bei den Stylusmodellen fallen diese Unterschiede mit 2,2% und 2,5% zwar geringer aus, zeichnen aber denselben Trend ab. In Tabelle 6.1 sind nochmals alle Ergebnisse numerisch und geordnet nach Konditionen dargestellt.

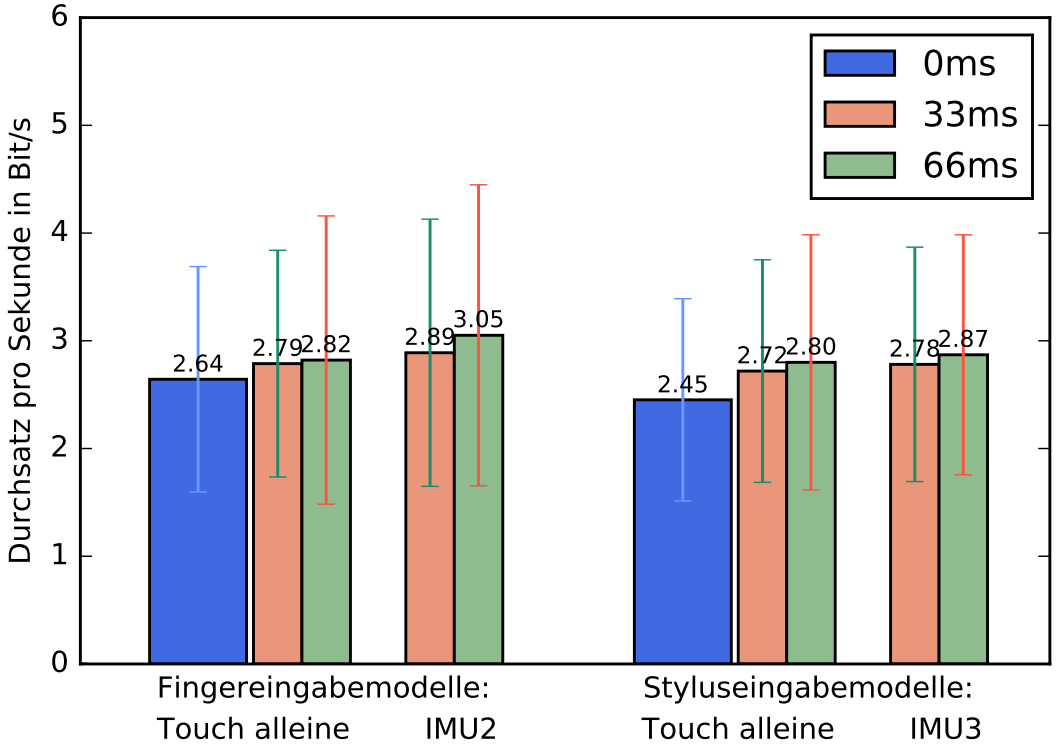


Abbildung 6.6: Die Abbildung stellt den durchschnittlichen Durchsatz bei der Bewältigung des Fitts' Law-Tests als Balkendiagramm in Bit/s nach MacKenzie dar [Mac92]. Die Balken wurden räumlich nach ihren vier zugehörigen Modellen angeordnet, welche darunter angeführt werden. Die Abkürzung IMU2 entspricht dem Modell Touch- und $IMU_{Handgelenk}$ für Fingereingaben und IMU3 entspricht dem Modell Touch- und IMU_{Stylus} für Stylusaufgaben. Die 0ms Konditionen wurden mit doppelter Breite dargestellt, aufgrund der doppelten Datengrundlage. Die Durchschnittswerte wurden an den Balken annotiert. Die Vorhersagestufen wurden farbkodiert. Die Fehlerbalken zeigen die Standardabweichung.

6.3.2 Varianzanalyse

Die im vorherigen Abschnitt gefunden Ergebnisse müssen noch auf signifikante Effekte untersucht werden. Dafür wurde hier, wie auch im letzten Kapitel, eine dreifaktorielle ANOVA, mit Bonferroni-Korrekturen für paarweise Vergleiche, auf den Durchschnittswerten der einzelnen Teilnehmer, durchgeführt. In Anlehnung an das gewählte Studiendesign handelt es sich um eine „repeated measures" ANOVA mit den unabhängigen Variablen Eingabemethode, IMU-Nutzung und Vorhersagestufe, welche alle Effekte, auf die Performanz der Nutzer, zwischen den vergleichbaren Konditionen des „Crossed and Nested"-Designs der Studie in Abbildung 6.1, vergleicht.

			Durchsatz($Bits/s$)		Aufgabenzeit(ms)		Genauigkeit(%)	
Eingabe	IMU-Nutzung	Stufe	Durchs.	SD	Durchs.	SD	Durchs.	SD
Finger	Touch alleine	0ms	2,643	1,046	678	207	97,5	2,1
		33ms	2,788	1,052	635	196	97,7	3,0
		66ms	2,821	1,338	662	260	94,9	5,0
	Touch + IMU2	33ms	2,889	1,240	636	219	98,4	1,6
		66ms	3,051	1,397	611	225	97,6	2,7
Stylus	Touch alleine	0ms	2,452	0,939	721	197	97,0	2,2
		33ms	2,719	1,033	654	208	98,0	3,0
		66ms	2,800	1,184	644	218	95,8	4,0
	Touch + IMU3	33ms	2,781	1,088	647	215	97,7	2,5
		66ms	2,870	1,114	615	200	97,2	1,7

Tabelle 6.1: In der Tabelle wurden die Fitts' Law-Durchschnittswerte mit Standardabweichungen (SD) für den Durchsatz, die Zeit für das Bewältigen einer einzelnen Fitts' Law-Aufgabe und die Genauigkeit dabei für die vier gewählten Modelle mit den zehn zugehörigen Konditionen dargestellt. Die einzelnen Konditionen werden dabei räumlich getrennt und über die drei unabhängigen Variablen Eingabe (Eingabemethode: Finger oder Stylus), IMU-Nutzung und Stufe (Vorhersagestufe: 0ms, 33ms, 66ms) identifiziert. IMU-Bezeichnungen wurden abgekürzt (IMU2 = $IMU_{Handgelenk}$, IMU3 = IMU_{Stylus}).

Dabei wurde ersichtlich, dass die Nutzung von IMUs einen signifikanten Effekt auf die durchschnittlichen Durchsätze der Anwender hatte [F(1,15) = 3,329, p = 0,023]. Ebenso gab es bedeutende Unterschiede zwischen den einzelnen Vorhersagestufen in Bezug auf die Durchsätze [F(2,90) = 2,233, p = 0,047]. Durch die Benutzung einer IMU konnten signifikante Differenzen zwischen der 0ms und der 66ms Vorhersage (p = 0,012), sowie zwischen der 33ms und der 66ms Vorhersage (p = 0,050) für Styluseingaben festgestellt werden. Ohne IMU gab es lediglich kennzeichnende Effekte zwischen der 0ms und der 66ms Vorhersage (p = 0,014) für Styluseingaben.

Bei Fingereingabemethoden mit IMU-Nutzung haben die paarweisen Vergleiche je zwischen 0ms und 33ms Vorhersagen (p = 0,022) und zwischen 0ms und 66ms Vorhersagen (p = 0,017) signifikante Effekte enthüllt. Bei den entsprechenden IMU-losen Konditionen gab es bedeutende Unterschiede zwischen der 0ms und der 33ms Vorhersage (p = 0,015) und der 33ms und 66ms Vorhersage(p = 0,022). Zwischen den Eingabemethoden, Finger und Stylus, selbst konnten keine signifikanten Effekte festgestellt werden [F(1,15) = 1,485, p = 0,242]. Durch die Einnistung der Vorhersagestufe in diesem Studiendesign konnten Interaktionseffekte nur zu den kreuzfaktorisierten Ebenen des Designs, nämlich der Kreuzung von Eingabemethode und IMU-Nutzung, verglichen werden und nicht zu den eingenisteten Vorhersagestufen, was keine signifikanten Effekte aufzeigte [F(1,15) = 1,518, p = 0,0237].

Auch bei den durchschnittlichen Zeiten zur Bewältigung einer einzelnen Fitts' Law-Aufgabe von den Studienteilnehmern konnten signifikante Unterschiede durch die Nutzung einer IMU nachgewiesen werden [$F(1,15) = 12{,}24$, $p = 0{,}003$], genauso wie kennzeichnende Unterschiede zwischen den Vorhersagestufen [$F(2,90 = 9{,}411$, $p < 0{,}001$]. Zwischen den Eingabemethoden [$F(1,15) = 1{,}018$, $p = 0{,}329$] und den Interaktionseffekten auf den Ebenen der gekreuzten Faktoren Eingabemethode und IMU-Nutzung [$F(1,15) = 0{,}0573$, $p = 0{,}461$] wurden keine signifikanten Merkmale festgestellt. Post-hoc-Signifikanztests zeigten für die durchschnittliche Aufgabenzeit ähnliche signifikante Unterschiede wie beim Durchsatz. Bei der durchschnittlichen Genauigkeit der Teilnehmer konnten weder durch IMU-Nutzungseffekte [$F(1,15) = 1{,}232$, $p = 0{,}284$], Eingabemethode [$F(1,15) = 0{,}024$, $p = 0{,}087$], noch durch die Vorhersagestufe [$F(2,90) = 1{,}301$, $p = 0{,}133$] oder durch Interaktionseffekte zwischen den gekreuzten Ebenen von IMU-Nutzung und Eingabemethode [$F(1,15) = 0{,}144$, $p = 0{,}389$] signifikante Unterschiede festgestellt werden.

6.3.3 Subjektives Feedback

Für das subjektive Feedback wurden die nicht-parametrischen Ergebnisse der sieben Punkte Likert-Skala [Lik32] herangezogen. Dabei haben Studienteilnehmer, für die zehn verschiedenen Konditionen, Bewertungen, für ihre subjektive Wahrnehmung von Jitter/Zittern, Latenzen/-Verzögerungen und die Unannehmlichkeit der Benutzung des Touchscreens, für Zeichen- oder Fitts' Law-Aufgaben, abgegeben. Nutzer wussten nicht, ob und welches Modell gerade im Hintergrund Vorhersagen trifft. Die Ergebnisse der Bewertungen sind in Abbildung 6.7 dargestellt, wobei reine Touchvorhersagen rot eingezeichnet wurden und IMU-verbesserte Vorhersagen in grün. Die Konditionen ohne Vorhersagen (0ms) wurden während der Studie je doppelt ausgeführt und die Daten anschließend zusammengefasst, weshalb die Datengrundlage dieser Ergebnisse doppelt so groß ist, wie die der anderen.

Wie erwartet war die Wahrnehmung von Jitter bei der Ausgangsbasis (0ms) sehr gering, da hier kein Jitter existierte. Ebenso wurde bei der Ausgangsbasis mehr Latenz verspürt, als bei den Vorhersagemodellen, was die Wirkung der Vorhersage auf die Wahrnehmung bestätigt. Die Wahrnehmung von Jitter und Latenz schien bei Zeichenaufgaben geringer zu sein, als bei Fitts' Law-Ziehaufgaben, was in etwa den Ergebnissen von Annett et al. entspricht [Ann+14]. Bei Fitts' Law-Aufgaben war die Unannehmlichkeit bei der Nutzung einer Vorhersage größer, als ohne eine Vorhersage.

Zur Evaluation wurde auf den Ergebnissen eine multifaktorielle ANOVA ausgeführt. Dafür mussten zuerst die nicht-parametrischen Daten mit „aligned rank transformations" (Abkürzung: ART) passend ausgerichtet werden, wie von Wobbrock et al. vorgeschlagen [Wob+11][1]. Dabei zeigten sich signifikante Effekte, auf die Wahrnehmung von Jitter, durch die drei unabhängigen Variablen Eingabemethode [$F(1,15) = 20{,}390$, $p < 0{,}001$], IMU-Nutzung [$F(1,15) = 51{,}759$, $p < 0{,}001$] und Vorhersagestufe [$F(2,90) = 245{,}966$, $p < 0{,}001$] des Studiendesigns. Auch ein

[1] ARTool-Werkzeug-Website: http://depts.washington.edu/madlab/proj/art/index.html

Abbildung 6.7: Zu sehen sind die Jitter-, Latenz- und Unannehmlichkeitsbewertungen der Teilnehmer, welche sie für die Aufgaben Zeichnen oder Fitts' Law mit den Eingabemethoden Finger oder Stylus abgegeben haben. Eine eins (Minimum) repräsentiert das beste Ergebnis, eine sieben (Maximum) das schlechteste. Die durchschnittliche Bewertung wurden gerundet über die Balken geschrieben. Die Balken wurden nach Vorhersagestufe (0ms, 33ms, 66ms) gruppiert. Rote Balken stellen reine Touchvorhersagen, grüne Balken stellen durch IMUs verbesserte Vorhersagen dar. Blaue Balken (0ms) haben die doppelte Breite aufgrund der doppelten Datengrundlage.

Interaktionseffekt zwischen den gekreuzten Ebenen vom Eingabemethode und IMU-Nutzung [$F(1,15) = 4{,}956$, $p = 0{,}041$] des Design konnte festgestellt werden. Bei der Zeichenaufgabe konnten ebenfalls signifikante Effekte auf die Wahrnehmung von Jitter durch Eingabemethode [$F(1,15) = 44{,}658$, $p < 0{,}001$], IMU-Nutzung [$F(1,15) = 9{,}382$, $p = 0{,}007$] und Vorhersagestufe [$F(2,90) = 96{,}721$, $p < 0{,}001$] festgestellt werden, aber es wurden keine Interaktionseffekte [$F(2,90) = 0{,}976$, $p = 0{,}339$] aufgedeckt.

Auf die Wahrnehmung von Latenz bei Fitts' Law-Aufgaben wurden Effekte durch IMU-Nutzung [$F(1,15) = 11{,}118$, $p = 0{,}004$] und verschiedene Vorhersagestufen [$F(2,90) = 31{,}435$, $p < 0{,}001$] nachgewiesen, jedoch keine Effekte durch die zwei verschiedenen Eingabemethoden [$F(1,15) = 0{,}067$, $p = 0{,}798$], sowie keine Interaktionseffekte von Eingabemethode $\times$ IMU-Nutzung [$F(1,15) = 0{,}156$, $p = 0{,}699$] des „Crossed"-Abschnitts des Studiendesigns. Analog ergaben sich dieselben Effekte für dieselben Faktoren IMU-Nutzung [$F(1,15) = 11{,}118$, $p = 0{,}004$] und Vorhersagestufen [$F(2,90) = 31{,}435$, $p < 0{,}001$] der passenden Zeichenaufgaben, sowie keine Effekte für Eingabemethode [$F(1,15) = 0{,}267$, $p = 0{,}612$] und keine Interaktionseffekte [$F(1,15) = 0{,}074$, $p = 0{,}789$]. Bei der wahrgenommenen Unannehmlichkeit konnten nur bei den Fitts' Law-Aufgaben Effekte der Vorhersagestufen [$F(2,90) = 33{,}193$, $p < 0{,}001$] gefunden werden, aber sonst keine Effekte von IMU-Nutzung [$F(1,15) = 0{,}605$, $p = 0{,}448$], Eingabemethode [$F(1,15) = 3{,}444$, $p = 0{,}083$] oder Interaktionseffekte [$F(1,15) = 2{,}832$, $p = 0{,}113$].

Auswertung der Interviews

Die Antworten der Studienteilnehmer auf die Interviewfragen wurden generalisiert, um die Ergebnisse zusammenfassen und analysieren zu können. Dabei kam heraus, dass 75% der Studienteilnehmer, demnach zwölf Teilnehmer, bereits Eingabelatenzen bei Touchbewegungen auf ihren eigenen Touchgeräten bemerkt hatten. Die restlichen Teilnehmer gaben an, nie darauf zu achten oder keine Latenz bemerkt zu haben. Von den ersteren zwölf hatten sieben Teilnehmer Latenz bei Scrollen, vier beim Spielen, weitere vier beim Schreiben oder Swypen[2] und eine Person beim Malen/Zeichnen bemerkt. Alle Teilnehmer, welche Latenzen bemerkt hatten, empfanden diese als störend, wobei vier sie als nur leicht störend deklarierten. Die restlichen acht wurden dagegen moderat durch bemerkbare Latenzen gestört. Die häufigste Antwort auf die Frage, welche Touchaufgaben von niedrigeren Latenzen profitieren würden, war Malen/Zeichnen mit acht Antworten. Darauf folgten Spielen mit sieben, Schreiben mit fünf, Scrollen mit vier, Tippen/Swypen mit vier, Schieben und Zoomen mit zwei und Ziehen mit einer Antwort. Auf die Frage, welche Vorteile sie sich von niedrigeren Latenzen erhofften, wurde achtmal mit höherer Geschwindigkeit, fünfmal mit höhere Genauigkeit, viermal mit weniger Frustration, zweimal mit besserer Konzentration und je einmal mit mehr Kontinuität beim Arbeiten und mehr Intuition geantwortet.

6.4 Diskussion

In dieser Studie wurden die Effekte untersucht, die Latenzreduzierungen durch Vorhersage von Touchpositionen, mit unterschiedlicher Vorhersagestufe und zusätzlicher IMU-Nutzung, auf die Wahrnehmung und Auswirkung von Latenz bei Nutzern haben. Dies wurde sowohl objektiv, als auch subjektiv untersucht. Die objektive Messung wurde mit der Messung von

[2] Swype-Website: http://www.swype.com/

Performanz, hauptsächlich als Durchsatz bei Fitts' Law-Aufgaben, durchgeführt. Die subjektive Messung erfolgte über Benutzererfahrungsbewertungen sowohl bei Fitts' Law-, als auch bei Zeichenaufgaben. Zusätzlich wurde qualitatives Feedback von den Studienteilnehmern über ein Interview eingeholt, welches die Relevanz der behandelten Thematik beleuchten sollte.

Durch eine Fehlplanung des Studiendesigns wurden die Konditionen der vergleichenden Ausgangsbasis (0ms Vorhersage), den Interaktionen ohne jegliche Vorhersage, mit beiden Eingabemethoden (Finger und Stylus), je doppelt durchgeführt. Dadurch entstanden doppelt so viele Ergebnisse für diese Konditionen, wie für die restlichen acht Konditionen. Die Ergebnisse wurden zusammengefasst, wodurch die Ausgangslage, aufgrund der doppelten Datengrundlage, besser gefestigt wurde. Dadurch ist die Sicherheit der Vergleiche mit den anderen Konditionen größer, da die Messung der Ausgangsbasis eine solide Messgrundlage besitzt.

In den Fitts' Law-Tests konnte ein erhöhter Durchsatz der Nutzer von bis zu 15% bei Fingereingaben und 17% bei Styluseingaben, bei der Nutzung von 66ms Vorhersagen, mit Verbesserung durch IMUs, nachgewiesen werden. Dabei traten keine negativen Effekte auf die Genauigkeit der Nutzer auf. Die gewählten IMU-Positionen aus Kapitel 1 haben, daraus folgernd, gut für alle Nutzer funktioniert. Die Vorhersagen ohne IMUs erzielten alle niedrigere Durchsätze als die mit IMUs, trotzdem waren die Durchsätze höher als ohne Vorhersage. Die gefundenen Effekte wurden alle anhand von ANOVAs bestätigt. Die Durchsatzergebnisse entsprechen auch in etwa denen von Jota et al., welche einen Fitts' Law-Ziehtest in ihrer Arbeit zur Durchsatzmessung verwendeten [Jot+13]. Die Ergebnisse von Henze et al., deren Arbeit als Vorlage für diesen Ansatz diente, waren für alle Konditionen höher, als die hier erhobenen Ergebnisse [HFS16]. Das könnte an verschiedenen Ursachen liegen. Zum einen an den Studienteilnehmern und ihren Qualifikationen, zum anderen an den Studienbetreuern und deren Verhalten. Bei Henze et al. wurden die Teilnehmer bei einer Fehlerrate von unter 5% ebenfalls dazu aufgefordert, schneller zu agieren. Je nachdem wie diese Aufforderung vom Studienbetreuer durchgesetzt wurde, kann dies die Ergebnisse beeinflusst haben. Weiterhin gab es bei Henze et al. weniger Durchläufe des Fitts' Law-Tests, wodurch Ermüdungserscheinungen weniger zum Tragen kamen. Abschließend war die Teilnehmergruppe in dieser Arbeit vielseitiger gemischt, als die von Henze et al. gewählten Teilnehmer, welche alle ähnliche Positionen, der gleichen Fakultät, der Universität Stuttgart belegten.

Die subjektiven Ergebnisse haben gezeigt, dass Vorhersagen allgemein die wahrgenommene Latenz reduzieren. Dies gilt sowohl für Fitts' Law-Aufgaben, als auch für Zeichenaufgaben. Ebenfalls konnte ein signifikanter Effekt durch die Nutzung von IMUs auf die wahrgenommenen Latenzen nachgewiesen werden. Vor allem bei Styluseingaben sind die Unterschiede bei der Wahrnehmung von Latenzen weitreichend. Da hier auch größere Verbesserungen bei den Fitts' Law-Durchsätzen gezeigt werden konnten, spricht das dafür, dass vor allem Styli durch die Hinzunahme von IMUs profitieren würden.

Jitter wurde dagegen bei allen Vorhersagen relativ stark wahrgenommen, jedoch bei Fitts' Law-Aufgaben stets stärker, als bei Zeichenaufgaben. Dieser signifikante Effekt könnte mit der Cursorgröße zusammenhängen, da das Kästchen, welches im Fitts' Law-Test bewegt wird, viel größer ist, als die Spitze einer Linie, welche bei Zeichenaufgaben gemalt wird. Dadurch

wirkt sich Jitter visuell stärker beim Fitts' Law-Test aus. Auch bei höheren Vorhersagestufen wird Jitter stärker wahrgenommen. Jitter wurde von den hier befragten Teilnehmern als sehr unangenehm befunden, weshalb vor allem die Fitts' Law-Aufgaben hohe Unannehmlichkeitsbewertungen erhielten. Trotz der erheblich schlechteren Bewertungen, aufgrund der höchsten Wahrnehmung von Jitter, hatten die 66ms Vorhersagemethoden die höchsten Durchsätze beim Fitts' Law-Test, was damit zusammenhängt, dass eine niedrigere Latenz einen größeren Einfluss auf die Performanz der Nutzer hat, als das räumliche Zittern [PS09].

Durch die qualitativen Antworten auf die Interviewfragen wurde klar, dass einige Touchscreennutzer Eingabelatenzen auf ihren eigenen Geräten bereits bemerkt hatten und diese als störend empfanden. Bei ihnen hatten sich Eingabelatenzen von Touchscreens bereits negativ auf ihre Performanz, ihre Geschwindigkeit, ihren Arbeitsfluss oder ihre Konzentration ausgewirkt. Aus ihrem Feedback war eine klare Beeinträchtigung der Benutzererfahrung durch Latenzen herauszulesen. Aufgrund ihrer weiteren Antworten und den hier gesammelten Ergebnissen lässt sich schlussfolgern, dass viele Anwendungen von geringeren Eingabelatenzen profitieren würden. Dazu gehören vor allem schnell reagierende Anwendungen, wie Spielen oder Malen, aber auch Anwendungen, welche eine hohe Präzision erfordern, wie etwa Auswählen oder Tippen. Anwendungen, die beides erfordern, wie zum Beispiel handschriftliches Schreiben oder Anwendungen, die nichts davon benötigen, wie etwa Scrollen, können ebenfalls von einer reduzierten Eingabelatenz profitieren. Da der Hauptgrund für die Unannehmlichkeit der vorgestellten Latenzkompensierungen Jitter war, würden vor allem Anwendungen ohne sichtbaren Cursor, zum Beispiel bestimmte Spiele oder Aktivitäten wie Scrollen, stark von den vorgestellten Modellen profitieren, da bei cursorlosen Aktivitäten der Jitter nicht für die Nutzer sichtbar ist. Davon ausgehend lässt sich die Relevanz der vorliegenden Arbeit als hoch einstufen.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Ansatz zur Kompensierung von Eingabelatenzen auf Touchscreens über Extrapolation der Eingabepositionen vorgestellt. Die Vorlage für diese Arbeit bildete der Ansatz von Henze et al., welche einen rein softwarebasierten Ansatz zur Latenzreduzierung vorstellten [HFS16]. In ihrem Ansatz trainierten sie künstliche neuronale Netze auf die Vorhersage von Touchtrajektorien durch Finger- und Styluseingaben. Für das Training nutzten sie, über eine Mal- und Schreibanwendung gesammelte, Touchtrajektorien. Dadurch konnten sie die Performanz ihrer Studienteilnehmer bei Fitts' Law-Tests deutlich erhöhen. Der vorliegende Ansatz griff diese Arbeit auf und kombinierte sie mit der Nutzung von inertialen Messeinheiten (IMUs). Solche Messeinheiten werden oft in der Navigation verwendet, um etwa Lage, Beschleunigung oder Ausrichtung eines Objekts zu erhalten und kommen in vielen kommerziellen Smartgeräten vor, wie Smartphones, Smartwatches oder Smartpens. Auf diese Weise wurden die neuronalen Netze mit zusätzlichen Informationen, über die Handbewegungen der Touchscreennutzer, versorgt und konnten präzisere Vorhersagen zu den Eingaben treffen.

Im Laufe dieser Ausarbeitung wurde, zur Erreichung des gesetzten Ziels, zuerst ein Testsystem entwickelt, welches Inertialsensoren seriell zu einem Touchgerät verband, unter Berücksichtigung mehrere Kritikpunkte. Die externe Sensorik sollte dabei die Sensorik und das Verhalten typischer Smartgeräte (Smartwatches, Smartrings, Smartpens) simulieren. Dafür wurden passende IMUs an den drei Positionen Handgelenk, Finger und an einem Touchstylus befestigt, an welchen normalerweise die Smartgeräte sitzen würden. Die inertialen Messdaten wurden anschließend hochfrequent an das Touchgerät gesendet, auf welchem die Latenz reduziert werden sollte. Der finale externe Sensorprototyp konnte kontinuierlich IMU-Daten mit einer Frequenz von etwa 600Hz zur Verfügung stellen. Dieses System wurde in einer anschließenden Studie verwendet, um Touch- und Sensordaten von 18 Studienteilnehmern zu sammeln.

Auf Grundlage der gesammelten IMU- und Touchdaten wurden zehn unterschiedliche neuronale Netze trainiert, welche als Modelle die Trajektorien von Toucheingaben vorhersagen sollten. Die Modelle nutzten je entweder Finger- oder Styluseingaben und konnten Toucheingaben in 33ms, 66ms und 99ms Zukunft vorhersagen. Alle wurden auf der selben Architektur und auf denselben erhobenen Touchdaten erstellt und trainiert, jedoch mit unterschiedlicher Nutzung von IMUs und ihren Messdaten. Zum Vergleich der Modelle wurden die euklidischen Distanzen zwischen den vorhergesagten Positionen und den realen Touchpositionen, für einen Satz nutzerunabhängiger Testdaten, für jedes Modell, berechnet. Dabei ergab sich, dass die neuronalen Netze, welche zusätzlich zu den Touchtrajektorien auch auf IMU-Daten trainiert

wurden, bis zu 30% genauere Vorhersagen auf den nutzerunabhängigen Testdaten erzeugten, als die entsprechenden, reinen Touchmodelle.

In einer zweiten Studie wurden vier der zehn trainierten Modelle ausgewählt, welche vergleichend an 16 Studienteilnehmern getestet wurden. Dabei wurden Modelle gewählt, welche zeitnah, über Smartwatches oder Smartpens, auch in aktuellen Geräten integriert werden können. Zusätzlich wurden die Modelle mit der Ausgangslage, der Eingabe ohne Vorhersage, verglichen. Als objektives Vergleichsmerkmal der Modelle wurde die Performanz der Nutzer gewählt. Dafür wurde der Durchsatz bei Fitts' Law-Aufgaben über alle Teilnehmer und alle Modelle gemessen und verglichen. Davon ausgehend konnte der Durchsatz der Nutzer mit einem IMU-Vorhersagemodell um 15% bei Fingereingaben und um 17% bei Styluseingaben gesteigert werden. Bei reinen Touchmodellen fiel die Performanzverbesserung signifikant geringer aus. Als subjektives Vergleichsmerkmal wurden Nutzerbewertungen zur Latenzwahrnehmung aufgezeichnet. Die subjektiv wahrgenommene Latenz der Studienteilnehmer konnte durch Vorhersagen signifikant reduziert werden. Vorhersagen mit IMU-Zusatz konnten die Wahrnehmung von Latenzen noch weiter reduzieren. Durch die Hinzunahme von IMUs in den Modellen konnte auch eine reduzierte Wahrnehmung von Jitter nachgewiesen werden, einem unerwünschten, visuellen Nebenprodukt von Vorhersagen. Die Ergebnisse wurden mit Varianzanalysen (ANOVAs) auf die Bedeutsamkeit der gefundenen Unterschiede überprüft.

Insgesamt wurde gezeigt, dass IMUs einen positiven Einfluss auf die Reduzierung von Latenzen haben. Durch sie konnten Vorhersagen zur Latenzkompensierung verbessert werden. Daraus entstanden geringere Wahrnehmungen von Latenzen und Jitter durch Nutzer, was die Benutzererfahrung verbesserte. Außerdem konnte die Performanz der Anwender, durch Hinzunahme von IMUs zu Vorhersagen, weiter verbessert werden.

Limitierungen und Ausblick

Beide Studien wurden mit einer kleinen, ausgewählten Gruppe von Teilnehmern durchgeführt, in welcher sich nur sehr wenige Stylusnutzer befanden. Für andere Personengruppen, mit anderem Nutzerverhalten, könnte dieser Ansatz demnach deutlich unterschiedliche Ergebnisse produzieren. Ebenso wurden für die Studien drei Aufgabentypen erstellt, welche nicht zwingend realen Interaktionen mit Touchscreens entsprechen müssen oder nicht alle möglichen Bewegungsmuster der typischen Touchscreeninteraktion aufspannen. Weiterhin handelte es sich bei beiden Studiendesigns um Laborstudien, in welchen Teilnehmer die Aufgaben im Sitzen durchführten, während das Touchgerät auf einem Tisch ruhte. Dadurch kann keine Aussage über die Performanz der Modelle oder das Verhalten der IMUs getroffen werden, wenn damit Aufgaben im Stehen oder Gehen durchgeführt werden. Um diese Limitationen zu umgehen, müssten demnach einige weitere Studien und Versuche mit vielen weiteren Personen durchgeführt werden. Weiterführend wurde nicht erforscht, wie der hier beschriebene Ansatz auf verschiedenen kommerziellen Geräten performt, was zum Beispiel Gegenstand einer folgenden Arbeit sein könnte.

In weiteren Arbeiten könnten auch größere Datensätze von mehr Benutzern gesammelt werden, um die Modelle weiter zu verbessern. Auch die richtige Verwendung von mehr Sensorik, wie etwa Magnetometerdaten oder die interne Sensorik des Touchgerätes, auf welchem die Latenzreduzierung stattfinden soll, könnten die Vorhersagen genauer machen. Schon eine stärkere Reduzierung des Jitters würde die Benutzerfreundlichkeit dieses Ansatzes stark verbessern. Dies kann beispielsweise durch mehr Sicherheit der Vorhersagen aufgrund einer größeren Datengrundlage geschehen. Portiert man diesen Ansatz auf kommerzielle Geräte, wie Smartwatches, welche Nutzer im Alltag mit sich führen, so könnte man kontinuierlich IMU- und Touchdaten erheben, um die Vorhersagen zu verbessern, Jitter zu reduzieren und das System besser an den Nutzer anzupassen, ähnlich wie virtuelle Tastaturen mit personalisierten, lernfähigen Lexikon. So würde man ein selbst verbesserndes System erhalten, welches die IMUs der bereits vorhandenen Nutzergeräte mobilisiert und damit lernt, die Eingabelatenzen von Touchscreens der Nutzer im Hintergrund zu verbessern. Die Portierung auf kommerzielle Geräte könnte bereits durchgeführt werden. In dieser Arbeit wurde eine serielle Verbindung zu den IMUs aufgebaut, um eine hochfrequente Abtastung der IMUs zu gewährleisten, aber auch IMUs von kommerziellen Smartgeräten, wir beispielsweise Smartwatches, können bereits hochfrequent abgetastet werden [LXH16]. Durch die zunehmende Verbreitung und Verbesserung von Smartwatches, Smartpens und zukünftig auch Smartrings [Mai17], bietet diese Arbeit einen bereits aktuell realisierbaren Ansatz für die zukünftige Kompensierung von Eingabelatenzen auf Touchscreens, ohne einen Bedarf an externer Hardware, abseits von bereits vorhandenen Smartgeräten der Nutzer.

Literaturverzeichnis

[AAA07] S. E. Alper, K. Azgin, T. Akin. „A high-performance silicon-on-insulator MEMS gyroscope operating at atmospheric pressure“. In: *Sensors and Actuators A: Physical* 135.1 (2007), S. 34 –42. ISSN: 0924-4247. DOI: http://doi.org/10.1016/j.sna.2006.06.043. URL: http://www.sciencedirect.com/science/article/pii/S0924424706004353 (zitiert auf S. 25).

[ADG11] G. Anderson, R. Doherty, S. Ganapathy. „User Perception of Touch Screen Latency“. In: *Design, User Experience, and Usability. Theory, Methods, Tools and Practice: First International Conference, DUXU 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I*. Hrsg. von A. Marcus. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 195–202. ISBN: 978-3-642-21675-6. DOI: 10.1007/978-3-642-21675-6_23. URL: http://dx.doi.org/10.1007/978-3-642-21675-6_23 (zitiert auf S. 14).

[AMR04] F. Ababsa, M. Mallem, D. Roussel. „Comparison between particle filter approach and Kalman filter-based technique for head tracking in augmented reality systems“. In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Bd. 1. 2004, 1021–1026 Vol.1. DOI: 10.1109/ROBOT.2004.1307284 (zitiert auf S. 15).

[Abd07] H. Abdi. „Bonferroni and Šidák corrections for multiple comparisons“. In: *Encyclopedia of measurement and statistics* 3 (2007), S. 103–107 (zitiert auf S. 71).

[Alb+09] A. Albarbar, A. Badri, J. K. Sinha, A. Starr. „Performance evaluation of MEMS accelerometers“. In: *Measurement* 42.5 (2009), S. 790 –795. ISSN: 0263-2241. DOI: http://doi.org/10.1016/j.measurement.2008.12.002. URL: http://www.sciencedirect.com/science/article/pii/S0263224108002091 (zitiert auf S. 23).

[All+01] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, J. E. Zacher. „Tolerance of temporal delay in virtual environments“. In: *Proceedings IEEE Virtual Reality 2001*. 2001, S. 247–254. DOI: 10.1109/VR.2001.913793 (zitiert auf S. 15).

[Ann+14] M. Annett, A. Ng, P. Dietz, W. F. Bischof, A. Gupta. „How Low Should We Go?: Understanding the Perception of Latency While Inking“. In: *Proceedings of Graphics Interface 2014*. GI '14. Montreal, Quebec, Canada: Canadian Information Processing Society, 2014, S. 167–174. ISBN: 978-1-4822-6003-8. URL: http://dl.acm.org/citation.cfm?id=2619648.2619677 (zitiert auf S. 14, 92).

[Ass16] D. Asselborn. „A Predictive Approach to Compensate Latency of Tangibles on Capacitive Multi-Touch-Displays“. Bachelor Thesis. RWTH Aachen University, 2016 (zitiert auf S. 16).

[Atma] *8-bit AVR Microcontrollers*. ATmega328/P. Atmel. Dez. 2016 (zitiert auf S. 34).

[Atmb] *8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller*. ATmega32U4. Atmel. Dez. 2016 (zitiert auf S. 33, 38, 39).

[Att] *Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash*. ATtiny85/V. Rev. 2586Q. Atmel. Aug. 2013 (zitiert auf S. 35).

[BB13] F. Bérard, R. Blanch. „Two Touch System Latency Estimators: High Accuracy and Low Overhead“. In: *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*. ITS '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, S. 241–250. ISBN: 978-1-4503-2271-3. DOI: 10.1145/2512349.2512796. URL: http://doi.acm.org/10.1145/2512349.2512796 (zitiert auf S. 14, 60).

[BJ17] A. Bianchi, S. Je. „Disambiguating Touch with a Smart-ring“. In: *Proceedings of the 8th Augmented Human International Conference*. AH '17. Silicon Valley, California: ACM, 2017, 27:1–27:5. ISBN: 978-1-4503-4835-5. DOI: 10.1145/3041164.3041196. URL: http://doi.acm.org/10.1145/3041164.3041196 (zitiert auf S. 74).

[Bie+99] R. Bierhals, K. Cuhls, V. Hüntrup, M. Schünemann, U. Thies, H. Weule. *Mikrosystemtechnik - wann kommt der Marktdurchbruch?: Miniaturisierungsstrategien im Technologiewettbewerb zwischen USA, Japan und Deutschland*. Technik, Wirtschaft und Politik : Schriftenreihe des Fraunhofer-Instituts für Systemtechnik und Innovationsforschung. Physica-Verlag HD, 1999. ISBN: 9783790812503 (zitiert auf S. 20).

[Btl] *Single-chip Bluetooth® low energy solution*. nRF8001. Ver. 1.3. Nordic Semiconductor. März 2015 (zitiert auf S. 31, 32).

[Büc06] C. Büch. „SPI–Serial Peripheral Interface“. In: *Physik-Seminar Universität Koblenz-Landau*. Bd. 27. 2006 (zitiert auf S. 35, 37).

[CG80] G. Campbell, S. Geller. „Balanced latin squares“. In: (1980) (zitiert auf S. 45, 46).

[CRCB15] E. Cattan, A. Rochet-Capellan, F. Bérard. „A Predictive Approach for an End-to-End Touch-Latency Measurement“. In: *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ITS '15. Madeira, Portugal: ACM, 2015, S. 215–218. ISBN: 978-1-4503-3899-8. DOI: 10.1145/2817721.2817747. URL: http://doi.acm.org/10.1145/2817721.2817747 (zitiert auf S. 14).

[Cat+15] E. Cattan, A. Rochet-Capellan, P. Perrier, F. Bérard. „Reducing Latency with a Continuous Prediction: Effects on Users' Performance in Direct-Touch Target Acquisitions“. In: *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ITS '15. Madeira, Portugal: ACM, 2015, S. 205–214. ISBN: 978-1-4503-3899-8. DOI: 10.1145/2817721.2817736. URL: http://doi.acm.org/10.1145/2817721.2817736 (zitiert auf S. 16, 18).

[DHS11] J. Duchi, E. Hazan, Y. Singer. „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization“. In: *J. Mach. Learn. Res.* 12 (Juli 2011), S. 2121–2159. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068 (zitiert auf S. 65).

[Deb+15] J. Deber, R. Jota, C. Forlines, D. Wigdor. „How Much Faster is Fast Enough?: User Perception of Latency & Latency Improvements in Direct and Indirect Touch“. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, S. 1827–1836. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702300. URL: http://doi.acm.org/10.1145/2702123.2702300 (zitiert auf S. 14, 15, 60, 78).

[Deb+16] J. Deber, B. Araujo, R. Jota, C. Forlines, D. Leigh, S. Sanders, D. Wigdor. „Hammer Time!: A Low-Cost, High Precision, High Accuracy Tool to Measure the Latency of Touchscreen Devices“. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. Santa Clara, California, USA: ACM, 2016, S. 2857–2868. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858394. URL: http://doi.acm.org/10.1145/2858036.2858394 (zitiert auf S. 7, 14).

[Du14] W. Y. Du. *Resistive, capacitive, inductive, and magnetic sensor technologies*. Series in Sensors. CRC Press, 2014. ISBN: 9781439812495 (zitiert auf S. 21, 23, 25, 27).

[EH01] A. El-Hoiydi. „Interference between Bluetooth networks-upper bound on the packet error rate“. In: *IEEE Communications Letters* 5.6 (2001), S. 245–247. ISSN: 1089-7798. DOI: 10.1109/4234.929601 (zitiert auf S. 34).

[Ell+97] S. R. Ellis, F. Bréant, B. M. Menges, R. H. Jacoby, B. D. Adelstein. „Operator interaction with virtual objects: effect of system latency“. In: *Advances in human factors/ergonomics* (1997), S. 973–976 (zitiert auf S. 15).

[FBL14] E. Freeman, S. Brewster, V. Lantz. „Tactile Feedback for Above-Device Gesture Interfaces: Adding Touch to Touchless Interactions“. In: *Proceedings of the 16th International Conference on Multimodal Interaction*. ICMI '14. Istanbul, Turkey: ACM, 2014, S. 419–426. ISBN: 978-1-4503-2885-2. DOI: 10.1145/2663204.2663280. URL: http://doi.acm.org/10.1145/2663204.2663280 (zitiert auf S. 74).

[Fit54] P. M. Fitts. „The information capacity of the human motor system in controlling the amplitude of movement.“ In: *Journal of experimental psychology* 47.6 (1954), S. 381. DOI: 10.1037/h0055392. URL: http://dx.doi.org/10.1037/h0055392 (zitiert auf S. 10, 50, 84).

[Flü10] H. Flühr. „Flugzeugsensoren“. In: *Avionik und Flugsicherungstechnik: Einführung in Kommunikationstechnik, Navigation, Surveillance*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 261–285. ISBN: 978-3-642-01612-7. DOI: 10.1007/978-3-642-01612-7_10. URL: http://dx.doi.org/10.1007/978-3-642-01612-7_10 (zitiert auf S. 20–22, 24, 25).

[GB10] X. Glorot, Y. Bengio. „Understanding the difficulty of training deep feedforward neural networks“. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Hrsg. von Y. W. Teh, M. Titterington. Bd. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, S. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (zitiert auf S. 66).

[Ga14] D. V. Großer, et al. „Trends in Future-Oriented Sensor Technologies“. In: *AMA Association for Sensor Technology: Berlin, Germany* (2014) (zitiert auf S. 19, 20, 22, 24, 25).

[Gün08] W. Günthner. *Enhancing Cognitive Assistance Systems with Inertial Measurement Units (Studies in Computational Intelligence)*. 1. Aufl. Springer Publishing Company, Incorporated, 2008. ISBN: 354076996X, 9783540769965 (zitiert auf S. 18).

[HFS16] N. Henze, M. Funk, A. S. Shirazi. „Software-reduced Touchscreen Latency“. In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '16. Florence, Italy: ACM, 2016, S. 434–441. ISBN: 978-1-4503-4408-1. DOI: 10.1145/2935334.2935381. URL: http://doi.acm.org/10.1145/2935334.2935381 (zitiert auf S. 9, 10, 17, 18, 48, 50, 60, 61, 72–74, 83, 95, 97).

[HM95] J. Han, C. Moraga. „The influence of the sigmoid function parameters on the speed of backpropagation learning“. In: *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings*. Hrsg. von J. Mira, F. Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, S. 195–201. ISBN: 978-3-540-49288-7. DOI: 10.1007/3-540-59497-3_175. URL: http://dx.doi.org/10.1007/3-540-59497-3_175 (zitiert auf S. 65).

[HP13] A. Holzinger, G. Pasi. *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data: Third International Workshop, HCI-KDD 2013, Held at SouthCHI 2013, Maribor, Slovenia, July 1-3, 2013, Proceedings*. Bd. 7947. Springer-Verlag Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-39146-0 (zitiert auf S. 18).

[HSE10] W. Hussy, M. Schreier, G. Echterhoff. *Forschungsmethoden in Psychologie und Sozialwissenschaften-für Bachelor*. Springer-Verlag, 2010. ISBN: 9783540959366 (zitiert auf S. 46).

[Hub87] O. Huber. *Das psychologische Experiment: Eine Einführung*. Hogrefe, vorm. Verlag Hans Huber, 1987. ISBN: 9783456842011 (zitiert auf S. 46).

[Hui+15] Y. Hui, T. Nan, N. X. Sun, M. Rinaldi. „High Resolution Magnetometer Based on a High Frequency Magnetoelectric MEMS-CMOS Oscillator“. In: *Journal of Microelectromechanical Systems* 24.1 (2015), S. 134–143. ISSN: 1057-7157. DOI: 10.1109/JMEMS.2014.2322012 (zitiert auf S. 27).

[Ito+16] Y. Itoh, J. Orlosky, K. Kiyokawa, G. Klinker. „Laplacian Vision: Augmenting Motion Prediction via Optical See-Through Head-Mounted Displays“. In: *Proceedings of the 7th Augmented Human International Conference 2016*. AH '16. Geneva, Switzerland: ACM, 2016, 16:1–16:8. ISBN: 978-1-4503-3680-2. DOI: 10.1145/2875194.2875227. URL: http://doi.acm.org/10.1145/2875194.2875227 (zitiert auf S. 15).

[Ivk+15] Z. Ivkovic, I. Stavness, C. Gutwin, S. Sutcliffe. „Quantifying and Mitigating the Negative Effects of Local Latencies on Aiming in 3D Shooter Games“. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, S. 135–144. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702432. URL: http://doi.acm.org/10.1145/2702123.2702432 (zitiert auf S. 15).

[JT14] B. F. Janzen, R. J. Teather. „Is 60 FPS Better Than 30?: The Impact of Frame Rate and Latency on Moving Target Selection“. In: *CHI '14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '14. Toronto, Ontario, Canada: ACM, 2014, S. 1477–1482. ISBN: 978-1-4503-2474-8. DOI: 10.1145/2559206.2581214. URL: http://doi.acm.org/10.1145/2559206.2581214 (zitiert auf S. 15).

[Jot+13] R. Jota, A. Ng, P. Dietz, D. Wigdor. „How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, S. 2291–2300. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481317. URL: http://doi.acm.org/10.1145/2470654.2481317 (zitiert auf S. 7, 15, 50, 95).

[KAB14] M. Krzywinski, N. Altman, P. Blainey. „Points of Significance: Nested designs“. In: *Nature methods* 11.10 (2014), S. 977–978. DOI: 10.1038/nmeth.3137. URL: http://dx.doi.org/10.1038/nmeth.3137 (zitiert auf S. 79).

[KB10] T. Kaaresoja, S. Brewster. „Feedback is... Late: Measuring Multimodal Delays in Mobile Device Touchscreen Interaction“. In: *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*. ICMI-MLMI '10. Beijing, China: ACM, 2010, 2:1–2:8. ISBN: 978-1-4503-0414-6. DOI: 10.1145/1891903.1891907. URL: http://doi.acm.org/10.1145/1891903.1891907 (zitiert auf S. 14).

[KR99] M. Kearns, D. Ron. „Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation“. In: *Neural Computation* 11.6 (1999), S. 1427–1453. ISSN: 0899-7667. DOI: 10.1162/089976699300016304 (zitiert auf S. 66).

[Käm+16] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, P. Hui. „Dissecting the End-to-end Latency of Interactive Mobile Video Applications“. In: *CoRR* abs/1611.08520 (2016). URL: http://arxiv.org/abs/1611.08520 (zitiert auf S. 14).

[LCR07] E. Lank, Y.-C. N. Cheng, J. Ruiz. „Endpoint Prediction Using Motion Kinematics“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, S. 637–646. ISBN: 978-1-59593-593-9.

DOI: 10.1145/1240624.1240724. URL: http://doi.acm.org/10.1145/1240624.1240724 (zitiert auf S. 16, 18).

[LXH16] G. Laput, R. Xiao, C. Harrison. „ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers“. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST '16. Tokyo, Japan: ACM, 2016, S. 321–333. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984582. URL: http://doi.acm.org/10.1145/2984511.2984582 (zitiert auf S. 75, 99).

[LaV+14] S. M. LaValle, A. Yershova, M. Katsev, M. Antonov. „Head tracking for the Oculus Rift“. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, S. 187–194. DOI: 10.1109/ICRA.2014.6906608 (zitiert auf S. 18).

[Lei+14] D. Leigh, C. Forlines, R. Jota, S. Sanders, D. Wigdor. „High Rate, Low-latency Multi-touch Sensing with Simultaneous Orthogonal Multiplexing“. In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: ACM, 2014, S. 355–364. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647353. URL: http://doi.acm.org/10.1145/2642918.2647353 (zitiert auf S. 16).

[Lik32] R. Likert. „A technique for the measurement of attitudes.“ In: *Archives of psychology* (1932) (zitiert auf S. 85, 92).

[MS03] I. S. MacKenzie, R. W. Soukoreff. „Phrase Sets for Evaluating Text Entry Techniques“. In: *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '03. Ft. Lauderdale, Florida, USA: ACM, 2003, S. 754–755. ISBN: 1-58113-637-4. DOI: 10.1145/765891.765971. URL: http://doi.acm.org/10.1145/765891.765971 (zitiert auf S. 49).

[MS95] S. T. March, G. F. Smith. „Design and natural science research on information technology“. In: *Decision Support Systems* 15.4 (1995), S. 251 –266. ISSN: 0167-9236. DOI: http://dx.doi.org/10.1016/0167-9236(94)00041-2. URL: http://www.sciencedirect.com/science/article/pii/0167923694000412 (zitiert auf S. 46).

[MW93] I. S. MacKenzie, C. Ware. „Lag As a Determinant of Human Performance in Interactive Systems“. In: *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. Amsterdam, The Netherlands: ACM, 1993, S. 488–493. ISBN: 0-89791-575-5. DOI: 10.1145/169059.169431. URL: http://doi.acm.org/10.1145/169059.169431 (zitiert auf S. 7, 8, 13, 15).

[Mac92] I. S. MacKenzie. „Fitts' Law As a Research and Design Tool in Human-computer Interaction“. In: *Hum.-Comput. Interact.* 7.1 (März 1992), S. 91–139. ISSN: 0737-0024. DOI: 10.1207/s15327051hci0701_3. URL: http://dx.doi.org/10.1207/s15327051hci0701_3 (zitiert auf S. 10, 50, 84, 88, 90).

[Maga] *3-Axis Digital Compass IC*. HMC5883L. Honeywell. Feb. 2013 (zitiert auf S. 30).

[Magb] *3-axis Electronic Compass*. AK8963. Asahi Kasei Microdevices. Okt. 2013 (zitiert auf S. 35, 39).

[Mai17] J. Maida. *Advances in Technology to Boost the Global Smart Rings Market Through 2021, Reports Technavio*. 2017. URL: http://www.businesswire.com/news/home/20170222005587/en/Advances-Technology-Boost-Global-Smart-Rings-Market (zitiert auf S. 10, 74, 99).

[Mar11] A. Marcus. *Design, User Experience, and Usability. Theory, Methods, Tools and Practice: First International Conference, DUXU 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part II*. Bd. 6770. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN: 978-3-642-21708-1. DOI: 10.1007/978-3-642-21708-1_30. URL: http://dx.doi.org/10.1007/978-3-642-21708-1_30 (zitiert auf S. 14).

[Mee+03] M. Meehan, S. Razzaque, M. C. Whitton, F. P. Brooks. „Effect of latency on presence in stressful virtual environments“. In: *IEEE Virtual Reality, 2003. Proceedings.* 2003, S. 141–148. DOI: 10.1109/VR.2003.1191132 (zitiert auf S. 15).

[Mpua] *MPU-6000 and MPU-6050 Product Specification*. MPU-6050. Rev. 3.4. InvenSense. Aug. 2013 (zitiert auf S. 30).

[Mpub] *MPU-9250 Product Specification*. MPU-9250. Rev. 1.1. InvenSense. Juni 2016 (zitiert auf S. 35, 36, 38, 39, 62).

[NH10] V. Nair, G. E. Hinton. „Rectified Linear Units Improve Restricted Boltzmann Machines“. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Hrsg. von J. Fürnkranz, T. Joachims. Omnipress, 2010, S. 807–814. URL: http://www.icml2010.org/papers/432.pdf (zitiert auf S. 65).

[Ned15] D. Nedelkovski. *MEMS Accelerometer Gyroscope Magnetometer and Arduino*. 2015. URL: http://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/ (zitiert auf S. 26, 27).

[Nel+00] W. T. Nelson, M. M. Roe, R. S. Bolia, R. M. Morley. *Assessing simulator sickness in a see-through HMD: Effects of time delay, time on task, and task complexity*. Techn. Ber. DTIC Document, 2000 (zitiert auf S. 15).

[Ng+12] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, P. Dietz. „Designing for Low-latency Direct-touch Input“. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST '12. Cambridge, Massachusetts, USA: ACM, 2012, S. 453–464. ISBN: 978-1-4503-1580-7. DOI: 10.1145/2380116.2380174. URL: http://doi.acm.org/10.1145/2380116.2380174 (zitiert auf S. 7, 8, 14, 16, 29).

[Nie17] M. Nielsen. *Using neural nets to recognize handwritten digits*. 2017. URL: http://neuralnetworksanddeeplearning.com/chap1.html (zitiert auf S. 64).

[Nir+15] S. Nirjon, J. Gummeson, D. Gelb, K.-H. Kim. „TypingRing: A Wearable Ring Platform for Text Input“. In: *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '15. Florence, Italy: ACM, 2015, S. 227–239. ISBN: 978-1-4503-3494-5. DOI: 10.1145/2742647.2742665. URL: http://doi.acm.org/10.1145/2742647.2742665 (zitiert auf S. 74).

[PG12] A. Pavlovych, C. Gutwin. „Assessing Target Acquisition and Tracking Performance for Complex Moving Targets in the Presence of Latency and Jitter“. In: *Proceedings of Graphics Interface 2012*. GI '12. Toronto, Ontario, Canada: Canadian Information Processing Society, 2012, S. 109–116. ISBN: 978-1-4503-1420-6. URL: http://dl.acm.org/citation.cfm?id=2305276.2305295 (zitiert auf S. 15).

[PS09] A. Pavlovych, W. Stuerzlinger. „The Tradeoff Between Spatial Jitter and Latency in Pointing Tasks“. In: *Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. EICS '09. Pittsburgh, PA, USA: ACM, 2009, S. 187–196. ISBN: 978-1-60558-600-7. DOI: 10.1145/1570433.1570469. URL: http://doi.acm.org/10.1145/1570433.1570469 (zitiert auf S. 17, 96).

[PS11] A. Pavlovych, W. Stuerzlinger. „Target Following Performance in the Presence of Latency, Jitter, and Signal Dropouts“. In: *Proceedings of Graphics Interface 2011*. GI '11. St. John's, Newfoundland, Canada: Canadian Human-Computer Communications Society, 2011, S. 33–40. ISBN: 978-1-4503-0693-5. URL: http://dl.acm.org/citation.cfm?id=1992917.1992924 (zitiert auf S. 15).

[PW14] P. T. Pasqual, J. O. Wobbrock. „Mouse Pointing Endpoint Prediction Using Kinematic Template Matching“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, S. 743–752. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557406. URL: http://doi.acm.org/10.1145/2556288.2557406 (zitiert auf S. 16, 18).

[Phi] *I²C-bus specification and user manual*. UM10204. Rev. 6. NXP Semiconductors. Apr. 2014 (zitiert auf S. 35, 36).

[Pos00] A Poston. „Human engineering design data digest“. In: *Washington, DC: Department of Defense Human Factors Engineering Technical Advisory Group* (2000) (zitiert auf S. 53).

[Rei89] H Reichl. „Aufbau-und Verbindungstechnik für Sensoren“. In: *tm-Technisches Messen* 56.JG (1989), S. 413–414. DOI: 10.1524/teme.1989.56.jg.413. URL: http://www.degruyter.com/view/j/teme.1989.56.issue-jg/teme.1989.56.jg.413/teme.1989.56.jg.413.xml (zitiert auf S. 20).

[Rud15] D. Ruddock. *Google Will Now Define "High Fidelity Sensor Support"For Android Devices, Has Extensive List Of Performance Requirements*. 2015. URL: http://www.androidpolice.com/2015/11/03/google-now-defines-high-fidelity-sensor-support-for-android-devices-has-extensive-list-of-performance-requirements/ (zitiert auf S. 47).

[SC05] R. H. Y. So, G. K. M. Chung. „Sensory Motor Responses in Virtual Environments: Studying the Effects of Image Latencies for Target-directed Hand Movement“. In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. 2005, S. 5006–5008. DOI: 10.1109/IEMBS.2005.1615599 (zitiert auf S. 15).

[SM04] R. W. Soukoreff, I. S. MacKenzie. „Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in {HCI}". In: *International Journal of Human-Computer Studies* 61.6 (2004). Fitts' law 50 years later: applications and contributions from human-computer interaction, S. 751 –789. ISSN: 1071-5819. DOI: http://doi.org/10.1016/j.ijhcs.2004.09.001. URL: http://www.sciencedirect.com/science/article/pii/S1071581904001016 (zitiert auf S. 16).

[SR13] P. Sedlmeier, F. Renkewitz. *Forschungsmethoden und Statistik: Ein Lehrbuch für Psychologen und Sozialwissenschaftler (2. überarbeitete und erweiterte Auflage)*. 2013 (zitiert auf S. 46).

[Sch11] T. Schäfer. *Statistik*. VS Verlag für Sozialwissenschaften, 2011. ISBN: 9783531169392. DOI: 10.1007/978-3-531-92446-5 (zitiert auf S. 46).

[Sch13] P. D. A. Schütze. „Punktsensoren: Messprinzipien, Sensortechnologien, Sensorelemente, optische Sensorik, AVT". In: *Sensorik für erneuerbare Energien und Energieeffizienz, Beiträge zum Workshop vom AMA Fachverband für Sensorik e.V.* (März 2013), S. 13–17 (zitiert auf S. 20).

[TH11] P. Tahmasebi, A. Hezarkhani. „Application of a Modular Feedforward Neural Network for Grade Estimation". In: *Natural Resources Research* 20.1 (2011), S. 25–32. ISSN: 1573-8981. DOI: 10.1007/s11053-011-9135-3. URL: http://dx.doi.org/10.1007/s11053-011-9135-3 (zitiert auf S. 64).

[TX15] P. Tsoi, J. Xiao. *Advanced touch input on ios*. Techn. Ber. Technical Report. Apple Inc, 2015 (zitiert auf S. 17).

[Tea17] D. D. Team. *Using Neural Networks With Regression*. 2017. URL: https://deeplearning4j.org/linear-regression (zitiert auf S. 64, 66).

[Vai07] P. P. Vaidyanathan. „The Theory of Linear Prediction". In: *Synthesis Lectures on Signal Processing* 2.1 (2007), S. 1–184. DOI: 10.2200/S00086ED1V01Y200712SPR003. URL: http://dx.doi.org/10.2200/S00086ED1V01Y200712SPR003 (zitiert auf S. 17).

[WF02] G. Welch, E. Foxlin. „Motion tracking: no silver bullet, but a respectable arsenal". In: *IEEE Computer Graphics and Applications* 22.6 (2002), S. 24–38. ISSN: 0272-1716. DOI: 10.1109/MCG.2002.1046626 (zitiert auf S. 18).

[Wob+11] J. O. Wobbrock, L. Findlater, D. Gergle, J. J. Higgins. „The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: ACM, 2011, S. 143–146. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1978963. URL: http://doi.acm.org/10.1145/1978942.1978963 (zitiert auf S. 92).

[Xia+14] H. Xia, R. Jota, B. McCanny, Z. Yu, C. Forlines, K. Singh, D. Wigdor. „Zero-latency Tapping: Using Hover Information to Predict Touch Locations and Eliminate Touchdown Latency“. In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: ACM, 2014, S. 205–214. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647348. URL: http://doi.acm.org/10.1145/2642918.2647348 (zitiert auf S. 16).

[Yaz16] M. Yazdani. *What is an intuitive explanation of the Xavier Initialization for Deep Neural Networks?* 2016. URL: https://www.quora.com/What-is-an-intuitive-explanation-of-the-Xavier-Initialization-for-Deep-Neural-Networks (zitiert auf S. 66).

[ZDB12] B. Ziebart, A. Dey, J. A. Bagnell. „Probabilistic Pointing Target Prediction via Inverse Optimal Control“. In: *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*. IUI '12. Lisbon, Portugal: ACM, 2012, S. 1–10. ISBN: 978-1-4503-1048-2. DOI: 10.1145/2166966.2166968. URL: http://doi.acm.org/10.1145/2166966.2166968 (zitiert auf S. 16, 18).

[Zie03] E. R. Ziegel. „The Elements of Statistical Learning“. In: *Technometrics* 45.3 (2003), S. 267–268. DOI: 10.1198/tech.2003.s770. URL: http://dx.doi.org/10.1198/tech.2003.s770 (zitiert auf S. 59).

Alle URLs wurden zuletzt am 26. 04. 2017 geprüft.

Abbildungsverzeichnis

Tabellenverzeichnis

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Place, Date, Signature