Modellierung und Erkennung dreidimensionaler Handschrift mittels Inertialsensorik

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Christoph Amma
aus Villingen-Schwenningen

Tag der mündlichen Prüfung:	10. 07. 2015
Erste Gutachterin:	Prof. Dr.-Ing. T. Schultz
Zweiter Gutachter:	Prof. Dr.-Ing. T. Asfour

Danksagung

Zuvorderst danke ich meinen Eltern, die mir mein Studium ermöglicht, mich stets unterstützt und mir dabei alle Freiheiten gelassen haben. Ich danke Kati Kemper für das Korrekturlesen der kompletten Arbeit. Weiter danke ich Prof. Tanja Schultz für die Betreuung, Unterstützung und Förderung während der Promotion. Auch der Austausch mit den Mitstreitern war stets sehr fruchtbar, hierfür danke ich Dirk Gehrig, der auch meine Diplomarbeit betreute, Michael Wand, Felix Putze, Dominic Heeger, Tim Schlippe, Thang Vu, Dominic Telaar, Matthias Jahnke, Christian Herff, Jochen Weiner und Marcus Georgi. Ebenso haben viele Studierende mit Ihren Arbeiten, sei es als Abschlussarbeit oder im Rahmen eines Hiwi-Jobs, zum Entstehen dieser Arbeit beigetragen. Am Anfang der einzelnen Kapitel sind die Studierenden erwähnt, die durch ihre Arbeiten wesentliche Beiträge geleistet haben. Für die Übernahme des Koreferats danke ich Prof. Tamim Asfour.

Zusammenfassung

Der Vision des *Wearable Computings* folgend, werden die Komponenten von Mensch-Maschine-Schnittstellen zunehmend direkt am Körper des Benutzers getragen. Dies hebt den derzeit durch die Verbreitung von Touchscreens vorherrschenden Dualismus von Ein- und Ausgabegerät auf. Während sich als favorisiertes Ausgabemedium Brillen mit integrierten, halbtransparenten Bildschirmen abzeichnen, ist die Frage nach dem geeignetsten Eingabemedium noch offen. Eine wesentliche Anforderung ist hier eine freihändige Bedienung, da erst diese eine nahtlose Integration in die Handlungen des Nutzers gestattet. Insbesondere die Eingabe von Text stellt hierbei eine zentrale Herausforderung dar.

Im Rahmen dieser Dissertation wird mit *Airwriting* eine Technologie präsentiert, die eine freihändige, jederzeit verfügbare, leicht erlernbare und effiziente Texteingabe für *Wearable Computing* Systeme erlaubt. Text kann dabei durch Schreiben in der Luft, ähnlich dem Schreiben auf einem virtuellen Notizblock, eingegeben werden. Für die Bewegungserfassung werden an der Hand oder am Handgelenk getragene Inertialsensoren verwendet.

In der Arbeit wird der Entwurf, die Implementierung und die Evaluierung eines Systems zur Erkennung von Handschrift mittels Inertialsensorik beschrieben. Dabei bleibt das Szenario nicht auf die Erkennung von in die Luft geschriebener Schrift beschränkt. Es wird auch die inertialsensorbasierte Erkennung traditioneller, mit einem Stift geschriebener, Schrift behandelt. In der Arbeit wird sowohl das Problem der Detektion als auch das der Erkennung behandelt. Detektion bedeutet, aus Alltagsbewegungen Handschriftsegmente herauszufiltern und Bewegungen, die keine Schrift darstellen, zu ignorieren. Dies ermöglicht eine Nutzung ohne explizite Start und Stop Kommandos, eine wesentliche Voraussetzung für die nahtlose Integration in alltägliche Handlungen. Im Falle der Erkennung besteht die Herausforderung darin, aus den gegebenen Sensorsignalen möglichst fehlerfrei den geschriebenen Text zu dekodieren. Schließlich wird die Texterkennung mit einer Gestensteuerung kombiniert, um eine vollständige Benutzerschnittstelle mit hohem Funktionsumfang zu realisieren.

Die wesentlichen Beiträge der Arbeit lassen sich wie folgt zusammenfassen:

- **Kontinuierliche Schrifterkennung mit Inertialsensoren**: Es wird erstmalig eine Architektur für die Erkennung kontinuierlicher Handschrift mit Inertialsensoren vorgestellt und evaluiert. Die Modellierung erfolgt dabei mit Hidden Markov Modellen auf Buchstabenebene. Der Ansatz wird sowohl auf Handschrift, die mit einem sensorbestückten Stift erfasst wird, als auch auf *Airwriting* übertragen. Dies eröffnet, neben der freihändigen Texteingabe durch Schreiben in der Luft, auch neue Anwendungsmöglichkeiten im Bereich der automatischen Erkennung von mit einem digitalen Stift geschriebener Schrift.
- **Modellierung auf Einzelstrichebene**: Basierend auf einer Menge primitiver Striche werden Buchstaben als Sequenzen von Strichen modelliert. Diese Modellierung erlaubt erstmalig die Erkennung von Zeichen oder Schreibvarianten, die in der Trainingsmenge nicht vorkommen. Zudem legt diese Modellierung die Grundlage für die effiziente Erkennung von frei definierbaren, beliebigen Symbolen und von Schriftsystemen, die eine sehr große Zahl an Zeichen besitzen, wie beispielsweise das chinesische Schriftsystem.
- **Airwriting-System**: Für die praktische Anwendbarkeit sind neben der Erkennung auch die ständige Verfügbarkeit und ein energieeffizienter Betrieb wesentlich. Um eine ständige Verfügbarkeit zu ermöglichen, wird ein Verfahren zur Detektion von Schrift eingeführt. Ein energieeffizienter Betrieb wird durch ein Verfahren zur Kompression der über Funk zu übertragenden Sensordaten ermöglicht. Aufbauend auf den entwickelten Methoden, wurde ein nahezu in Echtzeit funktionierender *Airwriting* Demonstrator implementiert, der sowohl die Detektion als auch die Erkennung kontinuierlicher Schrift auf einem großen Vokabular zeigt. Das System wurde vielfach ausgezeichnet und in der Öffentlichkeit präsentiert, unter anderem auf der CeBIT 2014.
- **Airactions - Gestensteuerung und Texteingabe**: Die Verbindung der Texteingabe via *Airwriting* mit einer Gestensteuerung wird anhand einer Beispielimplementierung einer neuartigen Benutzerschnittstelle gezeigt, welche die Erkennung von Kommandogesten und Airwriting zu einem neuen Interaktionsparadigma verbindet. Die Integration beider Technologien erlaubt erstmalig ein freihändiges Bedienkonzept für *Wearable Computing* Anwendungen, welches über einfache Steuerkommandos hinausgeht. Mit diesem Funktionsumfang lassen sich bereits alle wesentlichen Bedienelemente moderner Smartphones und Smartwatches abbilden.

Inhaltsverzeichnis

Kapitel 1

Einführung und Motivation

In diesem Kapitel wird das Konzept des Wearable Computing sowie dessen Konsequenzen für die Mensch-Maschine Interaktion erläutert. Davon ausgehend wird insbesondere auf das Problem der Texteingabe in mobile Rechnersysteme eingegangen. Der in dieser Dissertation verfolgte Ansatz wird vorgestellt und existierenden Ansätzen gegenübergestellt. Abschließend werden die wesentlichen Beiträge dieser Dissertation zusammengefasst und in das Forschungsgebiet eingeordnet.

1.1 Wearable Computing

Innerhalb weniger Jahre haben vernetzte Smartphones und Tablets unseren Umgang mit Computern und unsere Art zu kommunizieren verändert. Dienste, Informationen und Kommunikation sind jederzeit und überall verfügbar. Der Rechner wird vom reinen Werkzeug zum digitalen Begleiter. Dies ist ein erster Schritt in Richtung des Konzepts des *Wearable Computings*. Dieser von Steve Mann geprägte Begriff beschreibt die Vision des mit dem Menschen verschmelzenden Computers [Man97, SMR+97]. Rechner, Sensoren sowie Ein- und Ausgabegeräte werden am Körper getragen, entweder unter, über oder in der Kleidung integriert. Die Möglichkeit der Interaktion besteht permanent ohne explizite Aktivierung, und Handlungen des Nutzers werden nur so kurz wie nötig unterbrochen. Das Konzept des *Wearable Computing* zielt

darauf ab, Mensch und Technik zu verweben und direkt die Fähigkeiten des Menschen zu erweitern, statt den Rechner nur als Werkzeug anzusehen.

Smartphones sind eine erste, wenn auch noch rudimentäre, Instanz eines *Wearable Computers*. Das vorherrschende Interaktionsprinzip derzeitiger Geräte, vom Smartphone bis zum Tablet, ist allerdings durch den *Dualismus von Ein- und Ausgabegerät* vorgegeben. Die Ausgabe erfolgt über einen Bildschirm, der gleichzeitig via Berührung eine Selektion und Manipulation dargestellter Elemente erlaubt. Das Gerät muss also in jedem Fall in der Hand gehalten werden, sowohl zur Präsentation von Ausgaben als auch für Interaktionsaufgaben. Die logische Weiterentwicklung stellen Geräte dar, welche die Sinne des Menschen direkter ansprechen und eine für den Menschen natürlichere und immersivere Interaktion mit der digitalen Welt ermöglichen.

Auf Ausgabeebene sind das beispielsweise Brillen oder Kontaktlinsen, die Informationen direkt in das Sichtfeld des Nutzers einblenden und nahtlos in die visuelle Wahrnehmung unserer Umgebung integrieren [Sta13]. Möglich ist dies beispielsweise durch Brillen mit integrierten Bildschirmen, sogenannte Smart Glasses oder Augmented Reality Brillen. Letztere legen eine rechnergenerierte Ausgabeebene über das gesamte Sichtfeld und können damit die wahrgenommene Realität bei Bedarf mit Zusatzinformationen überlagern. Weiter ist auch eine auditive oder taktile Ausgabe möglich. Eingabegeräte und Interaktionsprinzipien für diese Geräte existieren bislang kaum und stellen die Entwickler vor große Herausforderungen. Das ideale Eingabeparadigma kommt ohne zu manipulierende Geräte aus, ist intuitiv zu nutzen, ohne visuelle Aufmerksamkeit bedienbar und sozial akzeptabel. Die beiden wesentlichen Modalitäten, die diese Kriterien erfüllen, sind Sprache und Gestik. Sprache ist die für Menschen natürlichste Form der Kommunikation und bietet sich damit als Interaktionsmodalität an. Allerdings hat sie als Eingabemethode auch in vielen Situationen eklatante Nachteile. So ist es in geräuschbehafteter Umgebung technisch immer noch schwierig, gesprochene Sprache akkurat zu erkennen. In leisen Umgebungen dagegen kann hörbare Spracheingabe störend wirken, beziehungsweise von Unbefugten mitgehört werden. Darüber hinaus sind räumliche Kommandos wie Objektauswahl oder Weiterblättern nur umständlich in Worte zu fassen. Gesten hingegen eignen sich sehr gut für räumliche Kommandos, sind durch Inertialsensorik unabhängig von Umgebungsbedingungen erfassbar und für Umstehende nicht störend. Beide Modalitäten ergänzen sich demnach aufgrund ihrer komplementären Eigenschaften, so dass zukünftige Schnittstellen voraussichtlich Sprache und Gestik kombinieren werden.

Die Eingabe von Text ist eine wesentliche Funktionalität einer Benutzerschnittstelle. Aufgrund der genannten Nachteile der Spracheingabe, beispielsweise bezüglich der Privatsphäre und sozialen Akzeptanz, ist Spracheingabe keine universelle Lösung zur Texteingabe. Daher stellt die Eingabe von Text eine zentrale Herausforderungen bei der Realisierung von Benutzerschnittstellen für *Wearable Computer* dar. Denn erst die Eingabe von Text ermöglicht über einfache Kommandos hinausgehende Funktionen wie die Eingabe von Suchtermini, Nachrichten, Notizen oder Namen. Eine einfache Möglichkeit zur Texteingabe in mobile Geräte stellen Tastaturen oder die mittlerweile dominanten Softtastaturen dar. Allerdings sind die beide Varianten nicht für *Wearable Computing* Anwendungen geeignet, da sie die Bedienung eines in der Hand zu haltenden Gerätes bei voller visueller Aufmerksamkeit erfordern. Die vorliegende Arbeit stellt eine Lösung für das Problem der Texteingabe vor, die ausschließlich auf Gesteninteraktion beruht. Text kann durch Schreiben in der Luft, ähnlich dem Schreiben auf einem virtuellen Notizblock, eingegeben werden. Diese Form der Eingabe wird im Weiteren auch mit *Airwriting* bezeichnet. Damit eröffnet sich die Möglichkeit eines neuartigen Interaktionsparadigmas, durch welches sich die Bedienung von *Wearables* nahtlos in Alltagshandlungen integrieren lässt.

1.2 *Airwriting* und *Airactions*

Beim *Airwriting* wird Text mit der Hand in die Luft geschrieben, als wäre die Hand ein Stift, mit dem man auf eine imaginäre Oberfläche schreibt. Die Orientierung der virtuellen Ebene im Raum ist dabei beliebig. Es kann also sowohl vor dem Gesicht, ähnlich dem Schreiben auf einer Tafel, als auch auf Hüfthöhe, ähnlich dem Schreiben in einen Notizblock, geschrieben werden. Die Größe der Buchstaben kann ungefähr im Bereich von 10 cm bis 20 cm liegen. Damit ist eine **unauffällige** Eingabe möglich, eine wesentliche Voraussetzung für die Nutzerakzeptanz. Der Text kann dabei **kontinuierlich ohne Pausen** eingegeben werden. Dabei werden die einzelnen Buchstaben übereinander geschrieben, das übliche räumliche Vorrücken nach rechts kann entfallen. Auch die Erkennung von ganzen Sätzen ist möglich und kann ohne Pausen zwischen den Wörtern erfolgen. Die Erfassung erfolgt über einen in einem Armband integrierten Inertialsensor, die **Hände bleiben immer frei**. Auch Arbeits- oder Freizeithandschuhe können problemlos getragen werden. Über eine automatische Detektion der Schriftbewegungen kann das System **ständig im Hintergrund aktiv** sein und muss nicht explizit aktiviert werden. Dies ermöglicht eine nahtlose Einbettung in andere Handlungen. Der

Nutzer schreibt große Druckbuchstaben des lateinischen Alphabets, es ist also **kein Lernaufwand** seitens des Nutzers nötig.

Inertialsensoren können allerdings auch benutzt werden, um die Schreibbewegungen eines herkömmlichen Stiftes zu erfassen. In der vorliegenden Arbeit wird gezeigt, dass auch darauf basierend eine Schrifterkennung möglich ist, obwohl die Bewegungen deutlich kleinräumiger sind. Dies ermöglicht die Verbindung klassischen Schreibens mit der digitalen Verfügbarkeit des Textes auf beliebigen Schreibmedien mithilfe eines Sensorstiftes. Derzeit ist hierzu zusätzlich zu einem Sensorstift auch eine spezielle Papierunterlage oder ein Spezialpapier notwendig.

Die in dieser Arbeit vorgestellten Experimente zur Schrifterkennung basieren auf lateinischer Schrift und nutzen den Vorteil, dass aus einer relativ kleinen Menge an Buchstaben ein großer Wortschatz gebildet werden kann. Für das chinesische Schriftsystemen, und damit in Teilen auch für das japanische, ist dies nicht der Fall. Hier existiert eine große Menge von Zeichen, die jeweils für sich selbst schon bedeutungstragend sind und die aus einzelnen Strichen zusammengesetzt sind. In dieser Arbeit wird gezeigt, dass eine Schrifterkennung auch auf Strichebene möglich und praktikabel ist und der Airwriting-Ansatz damit grundsätzlich auch auf andere Schriftsysteme übertragbar ist.

Neben Schrift können auch Hand- und Armgesten mit Inertialsensoren erfasst und erkannt werden. In Verbindung mit *Airwriting* eröffnet sich dadurch die Möglichkeit eines neuartigen Interaktionsparadigmas, durch welches sich eine Bedienung von *Wearables* ausschließlich mit Hand- oder Armbewegungen und Gesten realisieren lässt und keine Eingabegeräte in der Hand gehalten werden müssen. Insbesondere ist im Gegensatz zu Touchscreens keine oder zumindest nicht die komplette visuelle Aufmerksamkeit des Nutzers nötig. In dieser Arbeit wird unter dem Namen *Airactions* eine Proof-of-concept Implementierung eines solchen Bedienkonzeptes vorgestellt. Mittels Steuergesten können Eingaben bestätigt, durch Menüs navigiert oder Aktionen ausgelöst werden. Die Eingabe von Text erlaubt die schnelle Eingabe von Namen zur Auswahl von Kontakten, das Einfügen von Annotationen oder Notizen sowie die Eingabe von Suchtermini. Damit lässt sich ein bedeutend größerer Funktionsumfang realisieren, als dies mit einer kleinen Menge an Kommandogesten möglich ist.

Die Erkennung von Gesten und Schrift im Kontext von *Wearable Computing* Anwendungen bringt mehrere Herausforderungen mit sich. Die anvisierten Anwendungen verlangen eine Erkennung in Echtzeit bei gleichzeitig möglichst geringem Speicher- und Rechenaufwand. Batterielaufzeiten limitieren zudem die von den Sensoren über Funk übertragbaren Daten. Menschliche

Bewegungen unterliegen immer eine intraindividuellen und in bedeutend größerem Maße auch interindividuellen Varianz. Die Modellierung der Bewegungen muss dies entsprechend berücksichtigen. Für eine praktikable Texteingabe mittels Handschrift ist es zudem notwendig, ein großes Vokabular an erkennbaren Wörtern bereitzustellen.

In den folgenden Abschnitten werden zunächst existierende Ansätze für die Gestenerkennung und Texteingabe vorgestellt und in Bezug zur vorliegenden Arbeit gestellt.

1.3 Gestenerkennung mit Inertialsensorik

Hand- und Armgesten wurden bereits früh als mögliche Eingabemodalität für intuitive und natürliche Benutzerschnittstellen untersucht. Einfache Gesten können auf einfache Kommandos abgebildet werden, damit lassen sich bereits die Grundfunktionen einer Steuerung realisieren. Insbesondere im Bereich der automatischen visuellen Perzeption beschäftigt man sich schon lange mit der Erkennung von Gesten, um natürliche Interaktionsformen zu etablieren [YOI92, SP97, WH99, NS03, MA07]. Hier ist neben der Erkennung der Gesten auch die Identifikation und Verfolgung der Körperteile eine Herausforderung. Man unterscheidet generell statische, ortsgebundene und mobile Systeme. Die Stärken kamerabasierter Systeme liegen im stationären Einsatz, sie eignen sich aufgrund ständig wechselnder Umweltbedingungen aber weniger gut für mobile Anwendungen.

Im Bereich des *Wearable Computings* kommen daher häufig Inertialsensoren [HHH98, AAS+09, CAJ13] zum Einsatz. Unter dem Begriff der Inertialsensoren fasst man Beschleunigungs- und Drehratensensoren zusammen. Beschleunigungssensoren messen die translative Beschleunigung, die der Sensor erfährt, und Drehratensensoren, auch Gyroskope genannt, messen die Winkelgeschwindigkeit der Rotationsbewegung des Sensors. Damit lässt sich theoretisch die Bewegung des Sensors vollständig beschreiben, praktisch ist aber eine Rückrechnung der Bewegungstrajektorie aufgrund des Sensorrauschens nur für Zeitabschnitte von wenigen Sekunden möglich. Allerdings haben Hofmann et al. in [HHH98] gezeigt, dass auch direkt auf den Beschleunigungs- und Winkelgeschwindigkeitsdaten eine robuste Bewegungserkennung möglich ist. Die Sensoren sind zudem klein, energieeffizient, unempfindlich gegen Umwelteinflüsse und kostengünstig. Daher eignen sie sich gut für den Einsatz für die mobile Bewegungserfassung.

Die automatische Erkennung von Gesten auf Basis der Inertialsensorsignale erfolgt durch zwei Hauptkomponenten, die Detektion der Gesten in einem kontinuierlichen Datenstrom von Bewegungsdaten und die Erkennung der detektierten Geste. Die Erkennung von einzelnen Gesten bei bekanntem Start- und Endzeitpunkt kann in der Regel mit hoher Genauigkeit durchgeführt werden [HHH98, SPHB08, CAJ13]. Die Detektion von Gesten in kontinuierlichen Daten ist aufgrund der Variabilität menschlicher Bewegungen schwieriger und stark abhängig von der gewählten Gestenmenge und den Bewegungen, in welche die Gesten eingebettet sind. Das Vermeiden von falsch-positiven Erkennungen ist hierbei die derzeit größte Herausforderung [Sta14], da das unbeabsichtigte Auslösen von mit den Gesten verbundener Systemfunktionen im besten Fall ärgerlich für den Nutzer ist. Die Detektion von Gesten kann entweder explizit als Segmentierungsschritt vor der eigentlichen Erkennung [Pre08, AAS+09, RLNS10, PLH+11] oder implizit in Kombination mit der Erkennung erfolgen [LK99]. In der vorliegenden Arbeit wird ein hybrider Ansatz aus einem expliziten Segmentierungsschritt mit hoher Trefferquote und einer nachfolgenden impliziten Segmentierung zur Filterung der falsch-positiven Erkennungen verfolgt. Zusätzlich wird in Kapitel 7.2 ein speziell auf Schrift angepasster expliziter Segmentierungsschritt eingeführt.

1.4 Texteingabe für Computersysteme

Neben der Steuerung einfacher Funktionen stellt die Eingabe von Text einen wesentlichen Bestandteil einer Benutzerschnittstelle dar. Für mobile Anwendungen und *Wearables* lassen sich eine Reihe relevanter Kriterien zur Beurteilung der Eignung einer Texteingabemethode formulieren:

- Ständige unmittelbare Verfügbarkeit
- Freihändige Bedienung
- Kein Lernaufwand für den Benutzer
- Bedienbarkeit ohne visuelle Aufmerksamkeit
- Schnelle Texteingabe
- Soziale Akzeptanz

Da derzeit noch keine Methode alle diese Kriterien vollständig erfüllt, ist Texteingabe für Mobilgeräte und Wearables ein aktives Forschungsgebiet.

Im Folgenden werden verschiedene Ansätze zur Texteingabe vorgestellt und diskutiert, die über normale Tastaturen hinausgehen. Spracheingabe erfüllt viele der genannten Kriterien und wird zukünftig eine wichtige Rolle auf dem Feld der Texteingabe spielen. Die auf dem Gebiet der automatischen Sprachverarbeitung erzielten Fortschritte erlauben mittlerweile den Einsatz von Sprachtechnologie in Endprodukten für den Massenmarkt, wie beispielsweise "Google Now" und "Apple Siri". Wie bereits in Abschnitt 1.1 angemerkt, ist Spracheingabe aber in vielen Situationen aus Gründen der sozialen Akzeptanz unpassend. Im Folgenden wird daher ausschließlich auf alternative Methoden zur Spracheingabe eingegangen. Neben der Spracheingabe erfüllt der in dieser Arbeit präsentierte Ansatz des *Airwritings* die meisten der genannten Kriterien. Nur die Geschwindigkeit der reinen Texteingabe durch *Airwriting* bleibt hinter der Verwendung von tastaturbasierten Ansätzen zurück. Tastaturbasierte Ansätze können im Gegensatz zu *Airwriting* allerdings das zentrale Kriterium einer freihändigen Bedienung nicht erfüllen und eine ständige, unmittelbare Verfügbarkeit sowie eine Bedienbarkeit ohne visuelle Aufmerksamkeit ist mit tastaturbasierten Ansätzen schwierig umzusetzen. Zur sozialen Akzeptanz wurden bisher für keine der genannten Ansätze aussagekräftige Studien durchgeführt und auch der *Airwriting*-Ansatz wird im Rahmen dieser Arbeit nicht bezüglich dieses Kriteriums untersucht.

1.4.1 Bildschirmbasierte Eingabe

Aktuelle, auf Touchscreens basierende Systeme erlauben die Eingabe von Text im einfachsten Fall über Softtastaturen oder Minitastaturen. Sogenannte Gestentastaturen (englisch: Word Gesture Keyboards) erlauben eine effizientere Eingabe von Text auf regulären Qwerty Tastaturen [KZ04]. Um ein Wort zu schreiben, fährt der Nutzer die Buchstaben des Wortes auf der Tastatur nacheinander in einer kontinuierlichen Bewegung mit der Fingerspitze ab. Basierend auf der Trajektorie der Bewegung, wird anschließend das intendierte Wort erkannt. Auch in intelligenten Umgebungen (engl. Smartrooms) stellt die Texteingabe eine Herausforderung dar. Bildschirme sind hier meist in die Wände integriert und Nutzer stehen zur Interaktion mit der Umgebung nicht notwendigerweise direkt vor dem Bildschirm. Die Interaktion beruht in der Regel auf Arm- und Handgesten. Die entwickelten Techniken sind daher für die vorliegende Arbeit relevant und basieren teilweise auf den gleichen Interaktionsparadigmen. Das Konzept der Gestentastatur lässt sich auch auf aus der Distanz zu bedienende Bildschirme übertragen. Dies erlaubt eine sehr effiziente Texteingabe und benötigt so gut wie keinen Lernaufwand des Nutzers. Dies wurde sowohl von Markussen et. al [MJH14],

als auch in eigenen Arbeiten gezeigt [Mor11]. Für die Texteingabe auf Smartwatches stellt vor allem die Bildschirmgröße ein Problem dar. Hier werden beispielsweise mehrfach belegte Bildschirmtasten in Kombination mit einer prädiktiven Worterkennung [KD14] oder zoombare Qwerty Tastaturen vorgeschlagen [OHOW13]. Für alle diese Systeme ist allerdings ein Bildschirm mit direktem visuellen Feedback sowie die volle visuelle Aufmerksamkeit des Nutzers notwendig.

1.4.2 Tragbare Tastaturen und spezielle Handschuhe

Für mobile Anwendungen, in denen kein Touchscreen zur Verfügung steht, wurden eine Reihe teilweise sehr unterschiedlicher Ansätze zur Texteingabe vorgeschlagen und untersucht. Neben der auch in dieser Arbeit verfolgten gestenbasierten Schrifteingabe, welche in Absatz 1.5 detaillierter diskutiert wird, lassen sich zwei weitere Ansätze identifizieren: am Körper tragbare Tastaturen und spezielle mit Tasten versehene Handschuhe.

Bereits früh wurden Tastaturen mit reduziertem Layout entwickelt, die am Unterarm getragen werden können [MMB96]. Der Twiddler von Lyons und Starner [LSP+04, LPS04] ist eine Handtastatur, die einhändig, ohne die visuelle Aufmerksamkeit des Nutzers in Anspruch zu nehmen, bedient werden kann. Sie kann mittels einer Schlaufe an der Hand getragen werden und bietet eine Tastatur mit einem speziellen Layout. Damit ist die Tastatur immer verfügbar und kann unterwegs benutzt werden. Die Hand ist damit allerdings, auch während keine Texteingabe erfolgt, nicht mehr frei für andere Aufgaben. Das Schreiben auf dem Twiddler muss zudem erlernt werden. Eine weitere Kategorie stellen spezielle Handschuhe dar, die Eingaben über Taster an den Fingerspitzen ermöglichen. Bestimmte Zeichen werden durch die verschiedenen Kombination gleichzeitig gedrückter Tasten gegen eine harte Oberfläche [RS99] oder durch sequentielles Drücken verschiedener Finger gegen den Daumen [BMB12] eingegeben. Da spezielle Handschuhe getragen und ein völlig neuartiges Schreibsystem erlernt werden muss, erscheinen diese Ansätze allerdings nur für Spezialanwendungen praktikabel.

1.5 Schrifterkennung

Die Erkennung von *Airwriting*, also in die Luft geschriebene Schrift, stellt eine Möglichkeit dar, Text einzugeben, ohne dass dazu separate Geräte manipuliert werden müssen. Bisherige Ansätze zur Schrifterkennung in der Luft

lassen sich in zwei Kategorien einteilen: Systeme die Einzelbuchstaben erkennen und Systeme die kontinuierliche Schrift erkennen. Werden nur isolierte Buchstaben erkannt, ist für eine Worterkennung meist eine explizite Segmentierung der Buchstaben durch den Nutzer gefordert. Dies kann entweder über eine Bewegungspause einer bestimmten Länge oder einen Taster geschehen. Eine Erkennung von Einzelbuchstaben entspricht letztlich einer Gestenerkennung, wobei als Gestenmenge das Alphabet benutzt wird. Eine kontinuierliche Erkennung von Handschrift erfolgt hingegen ohne Pausen oder andere Segmentierungsaktionen zwischen Strichen, Buchstaben und Wörtern. Erst kontinuierliches Schreiben ermöglicht eine für den Nutzer intuitive und effiziente Eingabe von Text mittels Handschrift.

Häufig wird statt des regulären lateinischen Alphabets ein *unistroke* Alphabet verwendet. Ein solches Alphabet reduziert die Komplexität der Buchstaben und bildet sie auf möglichst einen einzelnen Strich ab. Das *unistroke* Alphabet ermöglicht so eine effizientere Eingabe und erhöht die Genauigkeit einer automatischer Erkennung [GR93]. Eine verbreitete Variante des *unistroke* Alphabets ist das ursprünglich in PalmOS eingesetzte Graffiti Alphabet. Der wesentliche Nachteil bei der Verwendung alternativer Alphabete ist, dass der Nutzer die Schrift erst erlernen muss.

Das in dieser Arbeit vorgestellte System zur Erkennung von *Airwriting* erlaubt kontinuierliches Schreiben mit dem regulären lateinischen Alphabet und erlaubt damit sowohl eine intuitive als auch effiziente Eingabe von Text. Es hebt sich damit von allen im Folgenden besprochenen, existierenden Ansätzen ab.

1.5.1 Alternative Sensortechnologien

Neben Inertialsensorik werden auch alternative Sensortechnologien, wie beispielsweise Kameras, eingesetzt. Diese werden in diesem Abschnitt diskutiert und darauf folgend die auf Inertialsensoren basierenden Ansätze in Abschnitt 1.5.2.

Eine Möglichkeit zur Erfassung von in die Luft geschriebener Schrift stellt die Erfassung mit Stereo- oder Tiefenkameras dar. Dies hat den Vorteil, dass in der Regel die Trajektorie der Hand direkt zur Verfügung steht und damit auch dem Nutzer ein direktes, visuelles Feedback über seine Bewegung gegeben werden kann. Allerdings müssen, neben der Erkennung von Gesten, die Videodaten verarbeitet werden und entsprechend die Hand identifiziert und verfolgt werden. Gerade für den mobilen Einsatz muss mit stark schwanken-

den Lichtverhältnissen, unterschiedlichen Hintergründen und Verdeckungen gerechnet werden, was die Verfolgung der Hand oder Finger schwierig macht. Neben Arbeiten mit am Körper getragenen Kameras werden, aufgrund der Nähe zur vorliegenden Arbeit, im Folgenden auch Ansätze mit stationären Kameras in intelligenten Umgebungen diskutiert.

Kristensson et al. [KNQ12] beschränken sich auf die Erkennung einzelner in die Luft geschriebener Graffiti-Zeichen, die mit einem stationären Kinect Sensor erfasst werden. Ni et al. stellen mit Airstroke [NBN11] ein auf Graffiti basierendes System zur Freihandtexteingabe vor. Die Bewegungen werden mit einer Kamera und einem speziellen Handschuh erfasst, der Nutzer segmentiert Buchstaben durch Fingergesten und eine Wortvervollständigung erhöht die Geschwindigkeit. Wir haben in [Mor11, SMA+12] gezeigt, dass die kamerabasierte, kontinuierliche Erkennung von Schrift mit lateinischem Alphabet möglich ist. Die Bewegungen des Armes werden dabei mit Stereokameras erfasst und die Trajektorie der Schrift in der Verlängerung des Armes auf den Bildschirm projiziert. Daniel Morlock hat diesen Ansatz in seiner Diplomarbeit [Mor11] mit Gestentastaturen und einer normalen, per Zeigegeste bedienbaren Tastatur verglichen. Die Eingabe mittels Gestentastaturen zeigte sich für diesen Einsatzzweck den anderen Varianten sowohl in der Benutzerzufriedenheit als auch der Effizienz überlegen. Bei Benutzerschnittstellen auf Basis von Bildschirmen mit direktem visuellen Feedback erscheint deshalb eine Texteingabe durch tastaturbasierte Methoden derzeit am vielversprechendsten.

Daneben existieren Ansätze mit am Körper getragenen Kameras zur Erfassung der Handbewegungen [BSLJ03, MMC09, LLJ06]. Allerdings beschränken sich diese auf die Erkennung einzelner Buchstaben des Graffiti-Alphabets [LLJ06] oder zielen auf die Erkennung von Gebärdensprache [BSLJ03] ab.

Neben Kameras erlauben auch kapazitive Sensoren die Erfassung von Bewegung. In diesem Fall geschieht das über die resultierende Verformung und geänderte Lage von Gewebeschichten [Rek01]. Cheng et al. zeigen in [CABL13] damit die Erkennung isolierter, in die Luft geschriebener, Buchstaben und Zahlen. Die Bewegungen werden über eine Manschette mit kapazitiven Sensoren am Handgelenk erfasst. Da die Handbewegungen nicht direkt, sondern über die Verformung des Gewebes am Handgelenk erfasst werden, kann derzeit noch keine Aussage über die Robustheit dieses Ansatzes unter realen Bedingungen getroffen werden.

1.5.2 Inertialsensorik

Die am häufigsten benutzte Sensormodalität für die Erkennung von in die Luft geschriebener Schrift sind Inertialsensoren. Auch in der vorliegenden Arbeit werden sie zur Bewegungserfassung eingesetzt. Eine Reihe von Arbeiten nutzen die in modernen Smartphones bereits integrierten Inertialsensoren, um eine darauf basierende Texterkennung zu realisieren [ACG+11, DKHR14]. Damit bieten sie eine potenzielle Alternative zu Softtastaturen. Der *PhonePoint Pen* von Agrawal et al. [ACG+11] nutzt die im Smartphone integrierten Beschleunigungssensoren, um in der Luft ausgeführte Schreibbewegungen des Nutzers zu erfassen und die Trajektorie zu rekonstruieren. Um die in Abschnitt 1.3 genannten Probleme der Trajektorienrekonstruktion zu lösen, muss der Nutzer Pausen zwischen jedem Einzelstrich machen. Ein kontinuierliches Schreiben ist im Gegensatz zu der in dieser Arbeit vorgestellten Methode nicht möglich. Einen ähnlichen Ansatz wie Agrawal verfolgen Deselaers et al. mit "GyroPen" [DKHR14]. Zum Schreiben wird dazu das Smartphone wie ein Stift in der Hand gehalten. Auf Basis der Gyroskopmessungen und eines Bewegungsmodells wird die Trajektorie rekonstruiert und anschließend mit einem herkömmlichen Handschrifterkenner erkannt. Allerdings muss sich der Nutzer an das Bewegungsmodell anpassen, was teilweise eine ungewöhnliche Art zu schreiben erfordert.

Bang et al. [BCK+03] benutzen ein mit Beschleunigungssensoren und Gyroskopen ausgestattetes stiftähnliches Gerät. Ähnlich wie beim *PhonePoint Pen* wird die Trajektorie rekonstruiert. Um die Fehlerakkumulierung klein zu halten, müssen Pausen zwischen einzelnen Strichen gemacht werden. Durch die Verwendung eines *unistroke* Alphabets muss in diesem Fall aber nur zwischen einzelnen Buchstaben pausiert werden. Aufbauend auf diesen Arbeiten modellieren Kim et al. in [KCK06] die An- und Absetzphasen des Stiftes mit einem Bayesnetz. Es werden hohe Erkennungsraten für die isolierte Erkennung von Groß- und Kleinbuchstaben erreicht, allerdings wird das Konzept nicht auf die kontinuierliche Erkennung ganzer Wörter oder Sätze übertragen.

Aufbauend auf die in der vorliegenden Arbeit vorgestellten Methodik beschreibt Chen in seiner Dissertation [Che13] die kontinuierlichen Erkennung von in die Luft geschriebenem Text mittels eines Wii Controllers. Die Nutzer müssen die Buchstaben allerdings mit einer vorgegebenen Reihenfolge der Striche ausführen. Dadurch wird die in 3.3.2 beschriebene intraindividuelle Varianz beim Schreiben von Großbuchstaben minimiert. Die Modellierung erfolgt mit Hidden-Markov-Modellen. Die Wörter werden sowohl auf Wortebene als auch auf Buchstabenebene modelliert. Eine personenunabhängige

Evaluierung auf einem Vokabular von 40 Wörtern liefert sehr geringe Fehlerraten. Das Grundsystem entspricht damit der von uns bereits in [AS12] vorgestellten Methodik und validiert damit den gewählten Ansatz. Die Bewegungen zwischen Buchstaben werden von Chen allerdings erstmalig abhängig vom Kontext, also dem vorangegangenen und nachfolgenden Buchstaben, modelliert. Das in der vorliegenden Arbeit vorgestellte System erlaubt im Gegensatz zur Arbeit von Chen die kontinuierliche Erkennung ganzer Sätze und nutzt ein unauffällig tragbares Armband. Die Variabilität beim Schreiben wird nicht künstlich ausgeschlossen, sondern eine Lösung zur Modellierung der Variabilität auf Buchstaben und Strichebene vorgestellt. Die verwendeten Vokabulare sind, mit einem Faktor von bis zu 200, weitaus größer. Darüber hinaus wird eine kontextabhängige Modellierung auf Einzelstrichebene vorgestellt.

1.6 Eigene Beiträge

Die Beiträge dieser Arbeit bestehen aus vier Hauptbestandteilen, die hier kondensiert dargestellt werden:

- Es wird erstmalig gezeigt, dass kontinuierliche Erkennung von Schrift mit Inertialsensoren möglich ist. Es wird eine Architektur eines Erkenners, basierend auf Hidden Markov Modellen, vorgestellt. Der Ansatz lässt sich sowohl auf Handschrift, die mit einem sensorbestückten Stift erfasst wird, als auch auf *Airwriting* übertragen. Damit eröffnen sich neue Möglichkeiten im Bereich der Schrifterkennung mit traditionellen Stiften als auch im Bereich der Eingabe von Text für *Wearable Computing*.

- Es wird eine Modellierung der Bewegung auf der Ebene einzelner Striche eingeführt und untersucht. Diese Modellierung erlaubt erstmalig die Erkennung von Zeichen oder Schreibvarianten, die in der Trainingsmenge nicht vorkommen. Zudem legt diese Modellierung die Grundlage für die effiziente Erkennung von Schriftsystemen, die eine sehr große Zahl an Zeichen haben, wie beispielsweise das chinesisch.

- Es wurde ein nahezu in Echtzeit funktionierender *Airwriting* Demonstrator implementiert, der sowohl die Detektion als auch die Erkennung kontinuierlicher Schrift auf einem großen Vokabular zeigt. Die Bewegungen werden mit einem unauffälligen Armband erfasst. Das Demonstratorsystem funktioniert ohne Lern- und Trainingsphase für beliebige

Nutzer. Damit wird die in dieser Arbeit vorgestellte Form der Texteingabe für Nutzer einfach erlebbar. Das System wurde vielfach ausgezeichnet und in der Öffentlichkeit präsentiert, unter anderem auf der CeBIT 2014.

- Mit *Airactions* wird eine Beispielimplementierung einer neuartigen Benutzerschnittstelle implementiert, welche eine Erkennung von Kommandogesten und Airwriting zu einem neuen Interaktionsparadigma verbindet. Die Integration beider Technologien erlaubt erstmalig ein freihändiges Bedienkonzept für *Wearable Computing* Anwendungen, welches über einfache Steuerkommandos hinausgeht.

1.7 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in neun Kapitel. Im ersten Kapitel wird die Arbeit in den größeren Anwendungszusammenhang eingebettet und motiviert. Im Zuge dessen werden auch alternative Ansätze zur Texteingabe und verwandte Arbeiten zur Gesten- und Schrifterkennung mit Inertialsensoren diskutiert. Im zweiten Kapitel werden notwendige Hintergrundinformationen zum Verständnis der Arbeit erläutert. Dies umfasst Grundlagen über die Motorik des Schreibens, eine Vorstellung der verwendeten Inertialsensorik und deren relevante Eigenschaften sowie eine Einführung in Hidden Markov Modelle und deren Anwendung in der Handschrifterkennung. Im dritten Kapitel werden sämtliche für diese Arbeit aufgenommenen Datensätze eingeführt. Für jeden Datensatz werden Zweck, verwendete Geräte, das Aufnahmeprotokoll und die Datenmenge angegeben. In Kapitel vier wird das Basissystem zur Erkennung von Handschrift mit allen Komponenten eingeführt. Auf dieses System bauen alle weiteren Kapitel auf. Im fünften Kapitel wird die Modellierung und Erkennung von Schrift mit einer Modellierung auf Buchstabenebene behandelt. Die experimentelle Evaluierung wird sowohl auf *Airwriting*-Daten als auch auf mit einem Sensorstift aufgenommenen Daten durchgeführt. In Kapitel fünf wird eine Modellierung auf Basis von Einzelstrichen eingeführt. Es wird gezeigt, dass auch mit diesem Ansatz eine Erkennung von Schriftzeichen möglich ist. Zudem wird der Einfluss des Kontextes, in dem ein Strich ausgeführt wird, erläutert und eine entsprechende kontextabhängige Modellierung vorgeschlagen und evaluiert. In Kapitel sechs wird das Demonstrator System vorgestellt. Im Zuge dessen wird die dafür nötige Schriftdetektionskomponente und eine Methode zur Komprimierung der Sensorsignale beschrieben. In Kapitel sieben wird das *Airactions*-System

vorgestellt, welches die Schrifteingabe mittels *Airwriting* mit einer Gestensteuerung kombiniert. Damit entsteht eine mobile, gestenbasierte Schnittstelle mit hohem Funktionsumfang. Die Komponente zur Gestendetektion und Erkennung wird beschrieben und evaluiert.

Kapitel 2

Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen zum Verständnis der Arbeit erläutert. Dies beinhaltet die eine Betrachtung der spezifische Motorik von Schreibbewegungen, die eine Erkennung von Schrift mit Inertialsensoren überhaupt erst erlaubt. Die Sensorik und die ihr zugrunde liegenden Prinzipien werden ebenfalls erklärt. Zuletzt werden die wesentlichen Algorithmen und Methoden eingeführt, die in dieser Arbeit benutzten werden.

2.1 Schreibmotorik

Schrift zeichnet sich in erster Linie durch ihre grafische Repräsentation aus und nicht durch die Bewegung, die sie produziert. Traditionelle Erkennungssysteme arbeiten daher auf der Trajektorie, also auf der über die Zeit entstehenden grafischen Repräsentation. Eine große Herausforderung stellen dabei die interindividuellen Unterschiede im Schriftbild dar. Während die meisten Schreiber innerhalb ihres eigenen Schriftbildes relativ konsistent sind, existieren große Unterschiede zwischen unterschiedlichen Schreibern.

Allerdings kann auch die gleiche grafische Schriftrepräsentation theoretisch über eine Vielzahl unterschiedlicher Bewegungen zustande kommen. Beispielsweise kann der Buchstabe N mit Pausen zwischen den einzelnen Strichen oder kontinuierlich geschrieben werden. Obwohl das grafische Resultat

das gleiche ist, unterscheiden sich die Bewegungen und damit der Verlauf der Geschwindigkeit und Beschleunigung. Eine automatische Erkennung von Schrift auf Basis der Beschleunigung und der Winkelgeschwindigkeit ist allerdings nur möglich, wenn ein Buchstabe durch konsistente, sich stark ähnelnde Bewegungen erzeugt wird. Dabei sind durchaus auch unterschiedliche Varianten pro Buchstabe möglich, jedoch müssen diese wiederum in sich konsistent sein und die Anzahl der Varianten muss endlich sein.

Schreibbewegungen routinierter Schreiber liegt nach [MM95] eine hochüberlernte Motorik zugrunde. So schreiben Mai und Marquardt: „Betrachtet man die Geschwindigkeits- und Beschleunigungsprofile wiederholt geschriebener Buchstaben oder Buchstabengruppen, so fällt die extrem hohe Wiederholgenauigkeit in der Bewegungsausführung auf“ [MM95, S. 10]. Sogenannte automatisierte Bewegungen in eine Richtung sind durch eingipflige und glatte Beschleunigungsprofile charakterisiert [MM95, S. 12]. Es findet also eine Bewegung mit genau einer Beschleunigungsphase und einer Abbremsphase statt. Jede Anforderung an den Schreiber, wie geschrieben werden soll, hat Einfluss auf die Bewegung. Denn „nicht erst die Instruktion, eine Figur nachzuzeichnen, sondern bereits kleine Änderungen der Schreibbedingungen wie der Versuch, besonders präzise zu schreiben, oder das Schreiben zwischen Linien können die automatisierte Ausführung einer Bewegung empfindlich stören. Eine Untersuchung von automatisierten Schreibbewegungen sollte deshalb dem Schreiber die größtmögliche Freiheit bieten, seine individuelle und gewohnte Handschrift zu erzeugen“[MM95, S. 15].

Die Invarianz des Beschleunigungsprofils einer Bewegung gilt nicht nur für Schriftbewegungen. Auch schnelle, strichförmige Bewegungen mit dem Arm weisen eine hohe Ähnlichkeit in der Beschleunigung der Hand bei mehrmaliger Ausführung auf ([DP09] zitiert nach [Mor81, SL81, VT80]). Damit erscheint sowohl die Erkennung von mit dem Arm in die Luft geschriebener Schrift als auch die Erkennung von Gesten, wie sie in Kapitel 8 vorgestellt wird, auf Basis von Inertialsensoren möglich. Die von uns aufgenommenen Daten bestätigen, dass die Invarianzen auch für mit dem ganzen Arm in der Luft ausgeführte Handschrift und Gesten gilt.

Abbildung 2.1 zeigt *Airwriting* Beschleunigungsprofile eines Buchstabens bei mehrfacher Ausführung. Der Verlauf des Beschleunigungssignals ist in zwei Raumrichtungen für zehn Wiederholungen des Buchstabens F dargestellt. Die Signale wurden mittels einer Kreuzkorrelation zeitlich so verschoben, dass sie die maximale Übereinstimmung aufweisen, da die Aufnahmen naturgemäß einen zeitlichen Versatz aufweisen. Es ist deutlich zu sehen, dass die Beschleunigungsprofile eine sehr große Ähnlichkeit aufweisen und weit-

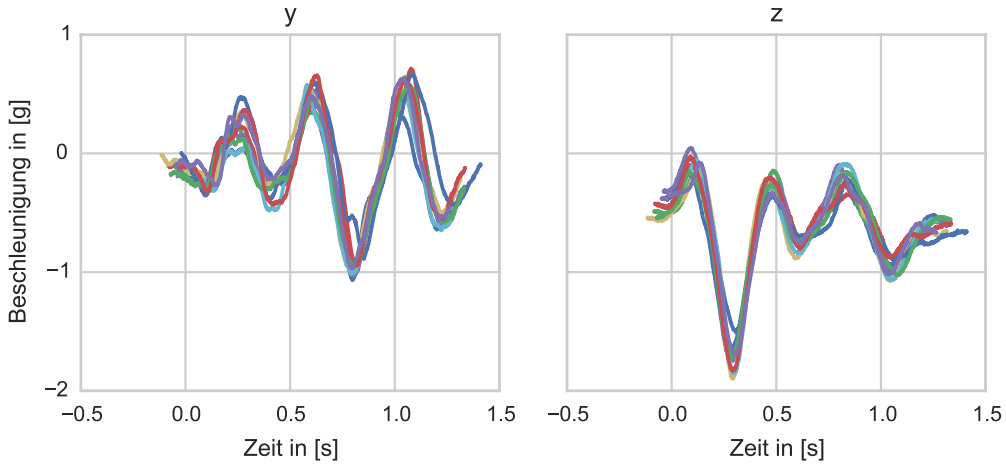


Abbildung 2.1 – Beschleunigungsverläufe während des Schreibens des Buchstabens F in die Luft. Jede Linie entspricht einer von 10 Wiederholungen derselben Person. Aus Darstellungsgründen werden nur der Verlauf in horizontaler (y, linke Abb.) und vertikaler Richtung (z, rechte Abb.) gezeigt.

gehend eingipflig und glatt sind. Die Daten wurden zeitlich nur verschoben und nicht skaliert. Demnach weisen die Bewegungen auch eine hohe zeitliche Synchronität zwischen den Wiederholungen auf. Eine Erkennung von in der Luft geschriebener Handschrift mittels Inertialsensoren scheint also aufgrund bestehender Erkenntnisse über die Motorik des Schreibens und erlernter Bewegungen sowie dem Abgleich mit aufgenommenen Daten grundsätzlich möglich. Diese Hypothese wird mit dieser Arbeit auch experimentell bestätigt.

Im Gegensatz zu traditioneller Schrift wird im Falle des *Airwritings* die Schreibbewegung mit dem ganzen Arm ausgeführt, da eine Bewegung aus dem Handgelenk mit einem Sensor am Arm nicht mehr erfassbar wäre. Der Sensor sitzt je nach benutztem Erfassungssystem am Handrücken oder am Handgelenk (siehe Kapitel 3). Geschrieben werden große Druckbuchstaben des lateinischen Alphabets. Die Höhe der imaginären Buchstaben liegt typischerweise im Bereich 10 cm bis 20 cm. Die Frequenz der Bewegung liegt im Falle von *Airwriting* im Bereich 2 Hz bis 5 Hz und variiert mit der Schreibgeschwindigkeit (siehe auch Abschnitt 7.2.1). Die Variation der Schreibgeschwindigkeit führt zu Unterschieden in der Amplitudenhöhe der Sensorsignale. In Abschnitt 4.2 wird eine Normalisierungsmethode zur Kompensation dieses Effekts eingeführt. Eine Gegenüberstellung der Sensorsignale für den

Fall des *Airwritings* und traditioneller Schrift ist in Abbildung 3.6 auf Seite 40 im Zuge der Beschreibung der Datensätze für den Buchstaben P gegeben.

2.2 Inertialsensorik

Als Inertialsensoren werden die folgenden zwei Arten von Sensoren bezeichnet:

- *Beschleunigungsensoren* messen die lineare Beschleunigung in einer Raumachse. Typischerweise liefern die Sensoren die Ergebnisse in g.
- *Drehratensensoren* oder *Gyroskope* messen die Winkelgeschwindigkeit um eine Raumachse in Grad pro Sekunde.

Es existieren eine Reihe unterschiedlicher Messprinzipien, die beispielsweise mechanisch oder optisch funktionieren. Für die Erfassung menschlicher Bewegungen kommen allerdings ausschließlich sogenannte MEMS (engl. microelectromechanical systems) Sensoren zum Einsatz. Diese Sensoren funktionieren nach einem mechanischen Messprinzip, werden allerdings, analog zur Herstellung integrierter Schaltkreise, direkt in Silizium über Lithographieverfahren hergestellt. Im Vergleich zu Inertialsensoren, die auf anderen Messprinzipien basieren, sind sie damit sehr klein und kostengünstig herstellbar. Die Abtastraten günstiger Sensoren liegen typischerweise zwischen 100 Hz und 1000 Hz.

Beschleunigungssensoren messen nie nur die Eigenbeschleunigung, sondern immer auch die Erdbeschleunigung. Diese geht als konstanter Wert von 1 g in die Messung mit ein und kann mit $9{,}81\,\mathrm{m/s^2}$ angenähert werden. Eine Subtraktion der Erdbeschleunigung von der Messung ist nur möglich, wenn die genaue Orientierung des Sensors im Raum bekannt ist. Abbildung 2.2 illustriert den Einfluss der Gravitation auf das Messergebnis, abhängig von der Sensororientierung. In Abschnitt 4.2 wird beschrieben, wie dieser Effekt im Rahmen der Signalverarbeitung kompensiert wird.

Die Kombination aus einem triaxialen Beschleunigungssensor und einem triaxialen Drehratensensor wird als Inertiale Messeinheit (englisch: *Inertial Measurement Unit (IMU))* bezeichnet. In der Theorie kann damit ein inertiales Navigationssystem implementiert werden, welches bei Bewegung des Sensors die Orientierungsänderung und die Trajektorie im Raum berechnet. Hierzu muss die Startposition bekannt sein. Dann kann über die einfache Integration der Winkelgeschwindigkeit die aktuelle Orientierung berechnet werden.

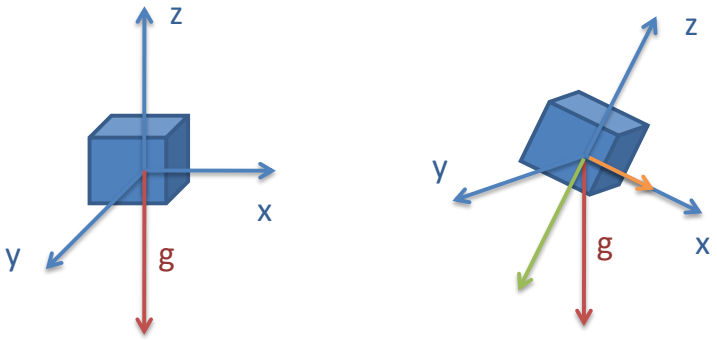


Abbildung 2.2 – Einfluss der Erdbeschleunigung auf die Messung, abhängig von der Orientierung des Sensors.

Anschließend kann die Erdbeschleunigung abgezogen werden und die Trajektorie über zweimalige Integration der gravitationsbereinigten Beschleunigung berechnet werden. Aufgrund der dreimaligen Integration akkumulieren sich Fehler schnell auf und innerhalb weniger Sekunden überwiegt der Fehler in der rekonstruierten Trajektorie [Woo07]. Durch die Verwendung von Magnetometern, welche die magnetische Flussdichte des Erdmagnetfelds messen, kann eine Bestimmung der Sensororientierung über eine externe Referenz erfolgen. Die Empfindlichkeit und Abtastrate reicht aber nicht an die der Drehratensensoren heran. Über einen Kalman-Filter, einen Komplementärfilter oder ähnliche Verfahren ist es möglich, die Sensorinformationen zu fusionieren und zumindest die Orientierung über die Zeit ohne großen Fehler zu verfolgen [MHV11]. Für eine robuste Rekonstruktion der Trajektorie sind die Sensoren und Verfahren allerdings nicht genau genug. Mehrere in 1.5.2 genannte Arbeiten zur Erkennung von Handschrift versuchen durch zusätzliche Modellannahmen eine Rekonstruktion von Handschrift zu realisieren und erreichen in einigen Fällen gute Ergebnisse. Allerdings funktionieren diese Rekonstruktionsverfahren nur, wenn der Nutzer seine Bewegungen an das System in hohem Maße anpasst. Weicht der Nutzer jedoch vom angenommenen Bewegungsmodell ab, funktionieren die Verfahren nicht mehr. Aus diesen Gründen wird in der vorliegenden Arbeit von einer Rekonstruktion der Trajektorie abgesehen.

2.3 Hidden Markov Modelle

In Abschnitt 2.1 wurde beschrieben, dass Schreibbewegungen einen charakteristischen zeitlichen Verlauf der Beschleunigung und Winkelgeschwindigkeit

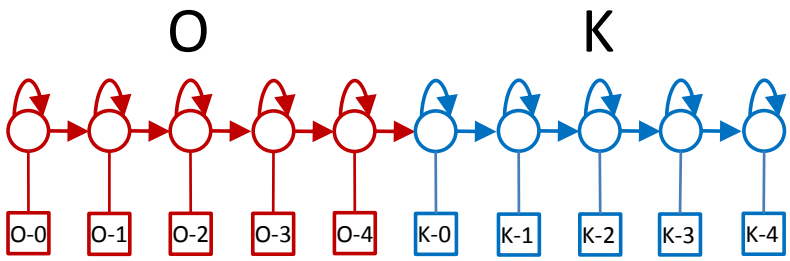


Abbildung 2.3 – Lineares links-rechts HMM für das Wort OK, bestehend aus den atomaren HMMs für O und K. Die Kreise symbolisieren die Zustände, die Pfeile die Transitionen und die Rechtecke die Verteilungen der Emissionswahrscheinlichkeiten.

haben. Dies bezieht sich entweder auf einen Stift oder, im Falle von Airwriting, auf die Hand. Die Zugehörigkeit zu einer Buchstabenklasse wird also über die charakteristische Zeitreihe der Beschleunigungs- und Winkelgeschwindigkeitswerte definiert. Hidden Markov Modelle, im weiteren Verlauf mit HMM abgekürzt, sind ein geeignetes Verfahren, um derartige Zeitreihen stochastisch zu modellieren. HMMs werden sehr häufig in der Handschrift- und Spracherkennung eingesetzt, da sich mit ihnen aus einer Reihe atomarer Modelle komplexere Modelle zusammensetzen lassen. So können beispielsweise aus Modellen einzelner Buchstaben in der Schrifterkennung oder einzelner Phoneme in der Spracherkennung Modelle für ganze Wörter konstruiert werden. Der Vorteil liegt nun darin, dass diese Wörter für die Erkennung *nicht* als Trainingsbeispiele vorliegen müssen, sondern ausschließlich Modelle für die atomaren Einheiten trainiert werden müssen.

Sowohl für das Training als auch die Erkennung existieren effiziente Algorithmen. Im folgenden Abschnitt werden HMMs formal eingeführt und die relevanten Algorithmen benannt. Die relevanten Verfahren zur Initialisierung, zum Training und zur Erkennung werden in ihrer Funktion grob erläutert. Auf eine umfassende Einführung in die HMM-Theorie und die verwendeten Algorithmen wird verzichtet. Es existiert ein umfangreicher Korpus an Literatur, auf den an den jeweiligen Stellen verwiesen wird. Falls spezielle Algorithmen verwendet werden, die in der Standard-Literatur nicht erwähnt werden, wird darauf hingewiesen. Eine allgemeine Einführung von HMMs findet sich beispielsweise in [B$^+$06, Kap. 13] oder in der Veröffentlichung von Rabiner [Rab89].

2.3.1 Aufbau und grundlegende Algorithmen

Ein HMM ist ein graphisches Modell und besteht aus einer Menge von Zuständen und Transitionen zwischen diesen Zuständen. Sie bilden eine Markov-Kette erster Ordnung, das heißt der Folgezustand ist ausschließlich vom derzeitigen Zustand abhängig. Die Zustände sind allerdings nicht direkt beobachtbar, sondern nur durch die von ihnen emittierten Beobachtungen, daher die Bezeichnung *hidden*. Die Beobachtungen sind als Sequenz von Beobachtungsvektoren $O = (o_1, \ldots, o_T)$ mit $o_i \in \mathbb{R}^D$ gegeben. Sie entsprechen den D-dimensionalen Merkmalsvektoren, die im vorliegenden Fall aus den Sensorsignalen berechnet werden. Die Beobachtungen werden als stationärer Zufallsprozess modelliert, dessen Verteilung vom aktuellen Zustand abhängt. Für jeden Zustand wird also eine Wahrscheinlichkeitsverteilung für die Beobachtungen angegeben. Formal lässt sich ein HMM entsprechend durch das Tupel (S, A, B, π) mit:

- die Menge der Zustände $S = S_1, \ldots, S_N$
- die $N \times N$ Matrix der Transitionswahrscheinlichkeiten A, wobei a_{ij} die Wahrscheinlichkeit angibt, von Zustand S_i in Zustand S_j überzugehen
- die Wahrscheinlichkeitsverteilungen B_i im Zustand S_i für $i \in 1, \ldots, N$ mit $B_i = P(O|S_i)$
- die Anfangswahrscheinlichkeitsverteilung π über alle Zustände

Es ergeben sich drei typische Probleme, für die jeweils effiziente Algorithmen zur Lösung existieren:

- Dekodierungsproblem: Gegeben eine Beobachtungssequenz, findet der Dekodierungsalgorithmus die wahrscheinlichste Zustandsfolge, die diese Beobachtungssequenz erzeugt. Die Berechnung kann mit dem Viterbi Algorithmus erfolgen.
- Lernproblem: Gegeben eine Beobachtungssequenz, findet der Lernalgorithmus diejenigen HMM Parameter (A, B, π), welche die Wahrscheinlichkeit zur Erzeugung der Beobachtungssequenz maximieren. Die Berechnung kann mit dem Baum-Welch Algorithmus, einer Instanz des EM-Algorithmus (englisch: expectation-maximization algorithm), erfolgen.
- Evaluierungsproblem: Gegeben eine Beobachtungssequenz, berechnet der Evaluationsalgorithmus die Wahrscheinlichkeit, dass diese Beob-

achtungssequenz vom HMM erzeugt wurde. Die Berechnung kann mit dem Forward Algorithmus erfolgen.

Für die Modellierung von Zeitreihen werden häufig sogenannte links-rechts oder lineare HMMs verwendet. Dabei besteht das Modell aus einer linearen Kette von Zuständen. Die Transitionen erlauben jeweils nur den Übergang in den nächsten Zustand oder das Verbleiben im aktuellen Zustand. Diese Topologie erzwingt jeden Zustand des Modells in einer vorgegebenen Reihenfolge zu durchlaufen. Sie eignet sich daher gut zur Modellierung kontinuierlicher Sensorsignale physikalischer Prozesse. Typischerweise werden in einem linearen HMM die Transitionswahrscheinlichkeiten alle auf 0,5 gesetzt. Damit ist der aktuelle Zustand ausschließlich von der beobachteten Signalsequenz abhängig. Eine lineare HMM-Topologie ist beispielsweise in Abbildung 2.3 dargestellt.

Mehrere Modelle lassen sich zu größeren Modellen zusammensetzen, indem die Zustandsmengen vereinigt und entsprechende Transitionen zwischen den Modellen hinzugefügt werden. Damit lassen sich aus einer Reihe atomarer Modelle beliebig komplexe, neue Modelle erstellen. Dies macht man sich sowohl in der Spracherkennung als auch in der Schrifterkennung zunutze. So können aus atomaren Buchstabenmodellen einfach Wortmodelle erstellt werden. Abbildung 2.3 stellt die Struktur eines HMM zur Modellierung des Wortes OK dar. Die atomaren HMMs für O und K sind zusammengefügt, um das Wort OK zu modellieren. Die atomaren Modelle für O und K bestehen in diesem Fall aus fünf Zuständen. Die Pfeile stellen die Transitionen zwischen den Zuständen dar. Bezogen auf Schriftbewegungen modellieren die Zustände einen kurzen Abschnitt der Bewegung mit einer Wahrscheinlichkeitsverteilung, welche die Verteilung der aus den Inertialsensordaten berechneten Merkmale in diesem Abschnitt modelliert. Eine größere Anzahl von Zuständen führt dazu, dass jeder einzelne Zustand einen kürzeren Abschnitt der Bewegung repräsentiert. Jeder Zustand konsumiert mindestens einen Merkmalsvektor, das heißt die Anzahl der Zustände muss kleiner als die Anzahl der beobachteten Merkmalsvektoren sein.

Die Modellierung der Wahrscheinlichkeitsverteilungen in den Zuständen erfolgt typischerweise durch Gaußsche Mischverteilungsmodelle (GMM) (siehe [B$^+$06, Kap. 9.2], da deren Parameter mit dem EM Algorithmus effektiv trainiert werden können und gleichzeitig beliebige Verteilungen modelliert werden können [B$^+$06, Kap. 9.2.2]. Eine Gaußsche Mischverteilung g besteht aus der gewichteten Summe von K Normalverteilungen mit

$$g(x) = \sum_{k=1}^{K} w_k \mathcal{N}_{\mu_k,\Sigma_k}(x) \ ,$$

wobei μ_k den Mittelwertvektor und Σ_k die Kovarianzmatrix der k-ten Normalverteilung bezeichnen. Für den Gewichtsvektor w muss

$$\sum_{k=1}^{K} w_k = 1$$

gelten, damit $g(x)$ eine Wahrscheinlichkeitsdichte ist. Da die Anzahl der zu schätzenden Parameter für die Kovarianzmatrix quadratisch mit der Dimensionalität des Merkmalsraums steigt, werden häufig, wie auch in dieser Arbeit, diagonale Kovarianzmatrizen verwendet.

2.3.2 Initialisierung und Training

Das Ziel der Initialisierung ist eine möglichst gute initiale Schätzung der Parameter der Gaußschen Mischverteilungen, also deren Mittelwerte und Kovarianzen. Hierfür existieren verschiedene Verfahren, die teilweise auch anwendungsspezifisch sind. In dieser Arbeit wird eine abgewandelte *Flatstart* (siehe [You08]) Prozedur durchgeführt. Im Falle eines *Flatstart* werden die Mittelwerte und Kovarianzmatrizen der Gaußschen Mischverteilungen einheitlich initialisiert. Um mit diesem Verfahren zu guten Ergebnissen zu kommen, müssen sehr viele Trainingsdaten vorhanden sein. Da im vorliegenden Fall relativ wenig Trainingsdaten vorhanden sind, werden die initialen Parameter der Gaußschen Mischverteilungen direkt auf den Trainingsdaten geschätzt. Da noch keine Zuordnung von Signalbereichen zu den Zuständen des HMM besteht, wird diese mit einer Gleichverteilung der Dauer über alle Zustände eines Modells abgeschätzt. Die Anzahl der Zustände des zu initialisierenden Modells sei mit $|S|$ gegeben. Für jedes Trainingsbeispiel wird die vorverarbeitete Sequenz der Merkmalsvektoren in $|S|$ gleich große Teile partitioniert. Die Merkmalsvektoren des n-ten Teils werden dem Zustand S_n zugeordnet. Für jeden Zustand werden die Merkmalsvektoren über alle Trainingsbeispiele akkumuliert. Anschließend werden die Daten geclustered, wobei die Anzahl der Mixturkomponenten die Anzahl der Cluster vorgibt. Die Mittelwerte und Varianzen der Cluster dienen der Initialisierung der Mixturkomponenten. Die Gewichte der einzelnen Komponenten werden alle auf den gleichen Wert gesetzt. Als Clusteringverfahren wird Neural Gas eingesetzt, eine robuster konvergierende Variante des k-Mittelwerte Algorithmus [MBS93].

Ziel des Trainings ist die Optimierung der GMM Parameter auf den Trainingsdaten. Als Optimierungskriterium wird das Maximum-Likelihood (ML) Kriterium verwendet. Klassischerweise wird hierfür der auf dem Forward-Backward Algorithmus basierende Baum-Welch Algorithmus verwendet. In

dieser Arbeit wird stattdessen der *Segmental K-Means* Algorithmus verwendet [JR90], der manchmal auch als Viterbi-Training oder als Baum-Viterbi Algorithmus bezeichnet wird. Über den Viterbi Algorithmus wird dabei der wahrscheinlichste Pfad durch das HMM für eine gegebene Merkmalssequenz gefunden. Anhand des Pfades können die einzelnen Merkmalsvektoren den Zuständen zugeordnet werden. Die Merkmalsvektoren werden für jeden Zustand über alle Trainingsbeispiele akkumuliert. Dann wird die Anpassung der GMM Parameter über den EM Algorithmus durchgeführt. Dieser Prozess wird iterativ mehrmals wiederholt. Der Unterschied zum Baum-Welch Algorithmus besteht also in der Verwendung des Viterbi Algorithmus statt des Forward-Backward Algorithmus. Im Gegensatz zum Baum-Welch Algorithmus ist der *Segmental K-Means* Algorithmus numerisch stabiler. Die Konvergenzeigenschaften sind dagegen vergleichbar [JR90, EM02].

2.3.3 Klassifikation

Bei der Klassifikation ist das Ziel, zu einer gegebenen Signalsequenz aus allen möglichen HMMs dasjenige zu finden, welches die höchste Wahrscheinlichkeit aufweist, die Signalsequenz zu generieren. Bei dem zu lösenden Problem handelt es sich um das in Abschnitt 2.3.1 eingeführte Evaluierungsproblem. Typischerweise wird hierfür der Forward Algorithmus verwendet. Dieser liefert, gegeben eine Merkmalssequenz und ein HMM, die Wahrscheinlichkeit, dass die Merkmalssequenz von diesem HMM generiert wurde. Eine schnellere Approximation liefert der Viterbi Algorithmus. Dieser berechnet den wahrscheinlichsten Pfad durch das HMM gegeben der Sequenz von Merkmalsvektoren. Dies kann als Approximation der Gesamtwahrscheinlichkeit verwendet werden. Insbesondere für die heuristische Suche in größeren Suchgraphen, wie in Abschnitt 4.7 beschrieben, ist eine Berechnung mit dem Forward Algorithmus im Kontext von Anwendungen im Bereich des *Wearable Computings* nicht mehr praktikabel.

2.4 Support Vector Maschinen (SVM)

Eine Support Vector Machine (SVM) ist ein binärer, linearer Klassifikator. Die Daten der zwei Klassen werden also durch eine Hyperebene getrennt. Dabei wird die Hyperebene so gewählt, dass der Abstand zu den ihr am nächsten liegenden Trainingsbeispielen der beiden Klassen maximiert wird. Damit wird eine eine möglichst gute Generalisierbarkeit erreicht [B$^+$06]. Da-

mit ist die SVM ein sogenannter *Large Margin* Klassifikator. Durch die Verwendung von Kernelfunktionen kann die SVM auf einfache Weise in einen nicht-linearen Klassifikator überführt werden [CV95]. Der Rechenaufwand für die Klassifikation selbst ist gering, womit sich SVMs für Echtzeitanwendungen eignen. Im Folgenden wird die grundlegende Idee der SVM zuerst für den Fall linear separierbarer Daten formuliert. Ausgehend davon wird die Behandlung nicht linear separierbarer Daten sowie die Verwendung von Kernelfunktionen eingeführt. Die Beschreibung orientiert sich an [B+06] und [Bur98]. Dort findet sich auch eine detaillierte Herleitung der Funktionsweise und der Eigenschaften von SVMs, auf die in dieser Arbeit verzichtet wird.

2.4.1 Linearer Fall

Gegeben seien eine Menge von linear separierbarer Trainingsvektoren $x_1, \ldots, x_N$ mit den entsprechenden Klassenzugehörigkeiten $t_1, \ldots, t_N$ und $t_n \in -1, 1$. Dann findet eine SVM eine Hyperebene

$$y(x) = w^T \phi(x) + b,$$

die die Trainingsdaten so trennt, dass $y(x_n) > 0$ für $t_n = 1$ und $y(x_n) < 0$ für $t_n = -1$, so dass $t_n y(x_n) = 1$ für alle Trainingsvektoren. Die Funktion ϕ ist eine Abbildung in den Merkmalsraum. Aus allen möglichen Lösungen für eine trennende Hyperebene wird diejenige gewählt, die den maximalen Abstand zu den Trainingsbeispielen hat. Damit bleibt ein möglichst großer Rand zwischen der Trennebene und den Daten. Das Finden dieser Ebene ist ein konvexes Optimierungsproblem mit Nebenbedingungen und kann daher effektiv mit Lagrange Multiplikatoren gelöst werden. Durch die Einführung einer Schlupfvariablen können auch linear nicht separierbare Daten und Ausreißer in den Trainingsdaten bearbeitet werden, indem erlaubt wird, dass Datenpunkte auf der falschen Seite der Trennebene liegen. Für die formale Einführung dieser Schlupfvariablen sei auf [B+06] verwiesen.

2.4.2 Nichtlinearer Fall

Für den nichtlinearen Fall macht man sich zunutze, dass das Optimierungsproblem in die duale Darstellung überführt werden kann, in der die Trainingsdaten nur als Skalarprodukt $\langle \phi(x), \phi(x') \rangle$ vorkommen. Das Skalarprodukt kann durch eine positiv definite Kernelfunktion $\kappa(x, x')$ ersetzt werden,

welche eine implizite Transformation des Merkmalsraums in einen höherdimensionalen, potentiell unendlichen Raum darstellt. In einem höher dimensionalen Merkmalsraum existieren trennende Hyperebenen und eine Menge nicht linear separierbarer Daten kann linear separierbar werden. Die konkrete Transformation ϕ in den höher dimensionalen Raum muss dabei weder während des Trainings noch während der Klassifikation neuer Daten explizit berechnet werden. Typische Kernelfunktionen sind beispielsweise radiale Basisfunktionen der Form

$$\kappa(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

mit σ als freiem Parameter. Die Optimierung der freien Parameter erfolgt in der Regel während des Trainings durch eine Gittersuche.

2.5 Auswahl der Verfahren

Neben den in den letzten beiden Abschnitten eingeführten Verfahren zur Sequenzmodellierung und Klassifikation existieren eine Vielzahl anderer Verfahren, die zur Lösung der beschriebenen Aufgaben geeignet sind. Im Folgenden sollen kurz die Gründe für die Auswahl der beschriebenen Verfahren und mögliche Alternativen aufgezeigt werden.

In der Spracherkennung wurden beispielsweise deutliche Fortschritte durch die Nutzung von Neuronalen Netzen statt Gaußschen Mischverteilungen in Kombination mit HMMs erzielt. Für das Training der Netze sind allerdings sehr große Datenmengen nötig, welche für das in dieser Arbeit beschriebene Problem nicht vorliegen.

Für binäre Klassifikationsprobleme, wie sie für die Detektion von Schrift und Gesten auftreten, existieren eine sehr große Anzahl verschiedener Klassifikationsverfahren. Für Support Vector Maschinen wurde in vielen Bereichen gezeigt, dass sie eine sehr gute Klassifikationsleistung erbringen. Zudem ist die Auswertung auch mit geringem Rechenaufwand durchführbar. Beide Verfahren, HMMs und SVMs, sind seit langem etabliert und es existieren effiziente Algorithmen und Softwarepakete für das Training und die Auswertung. Dies ist gerade im Hinblick auf die Anwendung im *Wearable Computing* relevant, da hier in der Regel nur eine verhältnismäßig schwache Rechenleistung in mobilen Geräten zur Verfügung steht. Insbesondere im Falle einer Sequenzmodellierung mit HMMs kann auf die jahrzehntelangen Entwicklungen und Fortschritte in der Algorithmik zur automatischen Spracherkennung zurückgegriffen werden.

2.6 Metriken für die Erkennung

2.6.1 Detektion

Die Detektion von Schrift und Gesten ist ein binäres Klassifikationsproblem, bei dem zwischen dem Vorkommen von Schrift oder einer Geste gegenüber der Abwesenheit von Schrift oder einer Geste im kontinuierlichen Sensordatenstrom unterschieden wird. Da die Klassenverteilung in den Daten in der Regel unbalanciert ist, werden die Genauigkeit (engl. Precision) und Trefferquote (engl. Recall) sowie deren harmonisches Mittel, das F-Maß (engl. F-Score), bestimmt. Seien mit r_p die richtig-positiven, mit r_n die richtig-negativen, mit f_p die falsch-positiven und mit f_n die falsch-negativen Ergebnisse der Klassifikation bezeichnet, dann berechnen sich die genannten Maße durch

$$\begin{aligned} \text{Trefferquote} &= \frac{r_p}{r_p + f_n} \\ \text{Genauigkeit} &= \frac{r_p}{r_p + f_p} \\ \text{F-Maß} &= \frac{2 \cdot \text{Trefferquote} \cdot \text{Genauigkeit}}{\text{Trefferquote} + \text{Genauigkeit}} \end{aligned}$$

2.6.2 Erkennung

Die Qualität der Schrift- und Gestenerkennung wird durch die Wortfehlerrate gemessen. Im Weiteren werden aufgrund ihrer Geläufigkeit die englische Abkürzungen WER (von Word Error Rate) verwendet. Die Metrik basiert auf der Levenshtein-Distanz zwischen der Referenz und der vom Algorithmus ausgegebenen wahrscheinlichsten Hypothese. Sie berechnet sich durch

$$\text{WER}_S = \frac{\sum_{i=1}^{k} S_{s_i} + I_{s_i} + D_{s_i}}{\sum_{i=1}^{k} N_{s_i}} \cdot 100,$$

wobei N_{s_i} die Anzahl der Wörter des Referenzsatzes s_i ist und S_{s_i} die Anzahl der Ersetzungen (engl. substitution), I_{s_i} die Anzahl der Einfügungen (engl. insertions) und D_{s_i} die Anzahl der Löschungen (engl. deletions) im Referenzsatz s_i bezeichnen. Das folgende Beispiel verdeutlicht die Berechnung:

```
Referenz:       we had a lot of expertise
Hypothese:      he had   lot of expert ease
Wortfehler:     S      D        S      I
```

Die Wortfehlerrate in dem Beispiel beträgt $(2 + 1 + 1)/6 \cdot 100 = 66\%$.

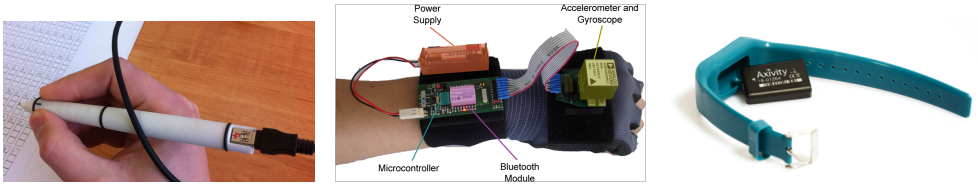
Kapitel 3

Datensätze

In diesem Kapitel werden die im Rahmen dieser Arbeit erstellten Datensätze eingeführt. Hierfür wird zuerst die verwendete Sensorhardware beschrieben. Danach wird ein Überblick über die Datensätze gegeben. Es werden Inhalt, Aufnahmeprozedur und Zweck der Datensätze beschrieben. Abschließend werden die Datensätze miteinander verglichen, Unterschiede benannt und Verfahren zur Kompensation dieser Unterschiede eingeführt.

3.1 Verwendete Hardware

Für die Aufnahme von Schrift- und Gestendaten wurden drei unterschiedliche mit Inertialsensorik ausgestattete Geräte verwendet: ein Stift, ein Handschuh und ein Armband. Für die Erfassung von Handschrift mit dem Stift wurde ein Prototyp eines mit Inertialsensoren ausgestatteten Stiftes verwendet. Bei dem Handschuh handelt es sich um einen selbst entworfenen und implementierten Sensorhandschuh zur Erfassung der Handbewegung, da es zum Zeitpunkt der Entstehung keine vergleichbaren kommerziellen Geräte gab. Durch die zunehmende Verfügbarkeit miniaturisierter Sensoren konnten weitere Aufnahmen mit einem in ein Armband integrierten Sensor durchgeführt werden. Abbildung 3.1 zeigt Fotos der drei Geräte. Im Folgenden werden die relevanten Eigenschaften der verschiedenen Hardware-Komponenten genauer beschrieben und gegenübergestellt.



(a) Stift **(b)** Handschuh **(c)** Armband

Abbildung 3.1 – Übersicht über die verwendeten Eingabegeräte: Prototyp des Sensorstiftes (a), selbst entwickelter Handschuh (b) und kommerzielles Armband (c).

Abbildung 3.1a zeigt den verwendeten Prototyp eines Sensorstiftes. Der Stift verfügt über eine herkömmliche Miene und ist zusätzlich mit Beschleunigungs- und Drehratensensoren ausgestattet. Ein Nutzer kann damit wie üblich auf Papier schreiben und zusätzlich die Bewegungsdaten des Stiftes aufzeichnen. Der Stift beinhaltet außerdem einen Drucksensor, welcher die Kraft misst, mit der die Mine auf das Papier gedrückt wird. Es findet keine Kalibrierung auf eine physikalische Größe statt, da im Rahmen dieser Arbeit ausschließlich die Rohdaten des Drucksensors benutzt werden. Die Datenübertragung erfolgt über ein Kabel, welches am Stiftende angeschlossen ist. Die Daten werden mit 100 Hz abgetastet.

Der in 3.1b abgebildete Handschuh wurde im Rahmen der Arbeit [Amm09] entwickelt. Der Bewegungssensor sitzt am Handrücken und kann damit Bewegungen aus dem Unterarm und dem Handgelenk erfassen. Als Sensor kommt ein Analog Devices ADIS16364 zum Einsatz, welcher Beschleunigung und Drehrate mit 819,2 Hz misst. Der Sensor liefert hochqualitative Daten und ist über einen integrierten Temperatursensor temperaturkalibriert. Die Datenübertragung erfolgt über Bluetooth.

Das in 3.1c abgebildete Armband mit Sensor wird von Axivity[1] vertrieben. Bei dem Sensor handelt es sich um das Modell WAX9. Beschleunigung und Drehrate sind einstellbar und wurden für die Erstellung der Datensätze mit 200 Hz abgetastet. Der Sensor beinhaltet neben einem Gyroskop und Beschleunigungssensor auch einen Barometer und einen Magnetometer, welche aber nicht benutzt wurden. Ein ebenfalls von Axivity vertriebenes Armband ermöglicht eine einfache Positionierung am Handgelenk. Der Sensor lässt sich damit wie eine Armbanduhr tragen. Entsprechend der Position des Sensors können ausschließlich Bewegungen des Unterarms erfasst werden. Die Datenübertragung erfolgt über Bluetooth.

[1] www.axivity.com

Datensatz	Arm-haltung	Inhalt	Probanden	Umfang Wörter	Umfang Zeichen
Stift	liegend		15		
Stift.G		Großbuchstaben	15		7830
Stift.K		Kleinbuchstaben	15		8100
Stift.W		Wörter	15	1350	9435
Handschuh	hoch		23		
Handschuh.B		Buchstaben	9		5850
Handschuh.W		Wörter	5	495	3695
Handschuh.S		Sätze	9	3294	16002
Strich			1		
Strich.P		Primitive	1		240
Strich.K		Kombinationen	1		120
Strich.S		Symbole	1		135
Strich.B		Großbuchstaben	1		390
Strich.Z		Zahlen	1		150
Detektion	frei	Sätze eingebettet in Alltagsaktivitäten	3	68	372
Armband			8		
Armband	frei	Sätze	8	2928	14224

Tabelle 3.1 – Überblick über die erstellten Schrift-Datensätze. Der Datensatz zur Gestenerkennung wird in Kapitel 8 gesondert beschrieben.

3.2 Beschreibung der Datensätze

Mit den drei Eingabegeräten wurden verschiedene Datensätze erstellt, welche nun detailliert vorgestellt werden. Sie dienen als Grundlage aller im Rahmen dieser Arbeit durchgeführten Experimente. Tabelle 8.1 gibt einen Überblick über die Datensätze, deren Inhalt und den jeweiligen Umfang. In einigen Fällen sind die Datensätze noch einmal gemäß ihres Inhalts in Unterdatensätze unterteilt. Für die Bezeichnung der Unterdatensätze wird an den Namen des Hauptdatensatzes ein durch einen Punkt getrenntes Kürzel angehängt. Der Stift-Datensatz wurde mit dem Sensorstift aufgenommen, der Handschuh-Datensatz mit dem Handschuh und der Armband-Datensatz mit dem Armband. Der Inhalt und das Aufnahmeverfahren der einzelnen Datensätze werden nun im Detail beschrieben. Der Datensatz, der im Rahmen der Experimente zur Gestenerkennung aufgenommen wurde, wird in Kapitel 8 zusammen mit dem speziellen Aufnahmesystem, der Auswahl der Gesten und den Experimenten beschrieben.

3.2.1 Der Stift-Datensatz

Der Stift-Datensatz wurde mit dem Sensorstift aufgenommen und gliedert sich in drei Unterdatensätze, die sich durch die Art des aufgenommenen Textes unterscheiden. Insgesamt 15 Probanden steuerten Aufnahmen zu den jeweiligen Datensätzen bei. Alle Probanden sind Rechtshänder. Jeder Proband führte eine Aufnahmesitzung durch, in der Daten für die drei Unterdatensätze aufgenommen wurden. Der Datensatz Stift.G enthält jeweils 18 Wiederholungen der 29 großen Druckbuchstaben des lateinischen Alphabets inklusive der deutschen Umlaute. Der Datensatz Stift.K enthält jeweils 18 Wiederholungen der 30 kleinen Druckbuchstaben des lateinischen Alphabets mit Umlauten und scharfem S. Der Datensatz Stift.W enthält 90 deutsche Wörter, die von jedem Probanden in großen Druckbuchstaben geschrieben wurden.

Für die Aufnahmen wurden Formularbögen vorbereitet, die von den Probanden auszufüllen waren. Die Bögen enthalten zeilenweise die zu schreibenden Buchstaben oder Wörter. Jeweils darunter sind leere Felder zum Beschreiben vorhanden. Die Felder hatten eine Höhe von 6 mm und die Probanden wurden angewiesen, in die Felder zu schreiben. Damit ist die Größe der Schrift zu einem gewissen Grad vorgegeben. Die Daten wurden immer in der gleichen Reihenfolge aufgenommen: erst die Großbuchstaben, dann die Kleinbuchstaben, danach die Wörter in Großbuchstaben. Die Einzelbuchstaben wurden ebenfalls immer am Stück in alphabetischer Reihenfolge aufgenommen. Auf eine Randomisierung der Reihenfolge wurde verzichtet, da Schreiben als ausreichend automatisierte Bewegung betrachtet werden kann, um von einer häufigen Wiederholung in gleicher Reihenfolge nicht stark beeinflusst zu werden (siehe Abschnitt 2.1). Mit der linken Hand segmentierten die Probanden über einen Tastendruck die Datenaufzeichnung der Buchstaben oder Wörter. Bei fehlerhafter Ausführung konnten die Probanden über die Aufnahmesoftware den Fehler markieren und die Eingabe wiederholen. Die Probanden wurden angewiesen, in Druckschrift, ansonsten aber so natürlich wie möglich, zu schreiben. Außerdem wurden sie angewiesen, den Stift so zu halten, dass sich die Markierung, die nahe der Stiftspitze angebracht ist, zwischen Daumen und Zeigefinger befindet. Hiermit wird sichergestellt, dass die Stiftdrehung um die Längsachse möglichst gering variiert. Eine Rotation des Stiftes führt zu einer Rotation des Sensorkoordinatensystems und macht damit den Vergleich der Daten schwierig.

Im Gegensatz zu den *Airwriting* Datensätzen enthält der Stift-Datensatz auch das Schriftbild der unterschiedlichen Probanden. Abbildung 3.2 zeigt Schriftproben aus dem Stift-Datensatz. Jede Zeile stammt dabei von einem anderen

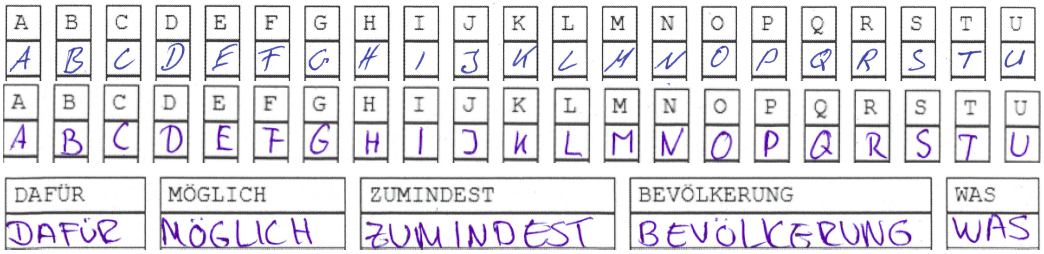


Abbildung 3.2 – Beispielschriftbilder aus dem Stift-Datensatz. Jede Zeile stammt von einem unterschiedlichen Probanden. Die Felder haben im Original eine Höhe von 6 mm.

Probanden. Die interindividuellen Unterschiede im Schriftbild sind deutlich zu sehen.

3.2.2 Der Handschuh-Datensatz

Der Handschuh-Datensatz beinhaltet Aufnahmen von insgesamt 23 Probanden. Der Datensatz besteht aus drei Unterdatensätzen. Der Datensatz Handschuh.B enthält Aufnahmen von einzelnen Buchstaben. Neun Probanden schrieben hierfür jeweils 25 mal die 26 Großbuchstaben des lateinischen Alphabets in Druckschrift. Der Datensatz Handschuh.W enthält Aufnahmen von kontinuierlich geschriebenen einzelnen Wörtern. Hierfür schrieben fünf Probanden jeweils 99 englische Wörter. Die Länge der Wörter ist zwischen zwei und elf Buchstaben gleich verteilt. Der Datensatz Handschuh.S enthält Aufnahmen kontinuierlich geschriebener einzelner Sätze. Insgesamt neun Probanden schrieben jeweils 80 englische Sätze.

Die Aufnahmeprozedur war für alle Unterdatensätze des Datensatzes Handschuh gleich. Die Probanden saßen auf einem Stuhl vor einem Laptop, auf dessen Monitor der zu schreibende Text dargestellt war. Durch einen Tastendruck starteten die Probanden die Aufnahme, durch einen weiteren Tastendruck stoppten sie die Aufnahme. Die Probanden sollten in der Luft schreiben, als ob sie auf eine imaginäre Tafel schreiben würden. Dabei waren sie angehalten, Druckbuchstaben von ca. 20 cm Höhe zu schreiben. Bei Wörtern und Sätzen wurden die Probanden angewiesen, die aufeinander folgenden Buchstaben jeweils an der gleichen Position zu schreiben und nicht wie während des Schreibens auf Papier fortlaufend nach rechts. Bei einem Fortschreiben nach rechts erfordert das Schreiben einzelner Wörter mittlerer Länge im Sitzen eine Körperdrehung, beziehungsweise im Stehen eine seitliche Bewegung. Eine Bewegung des ganzen Körpers ist in der Anwendung aber nicht

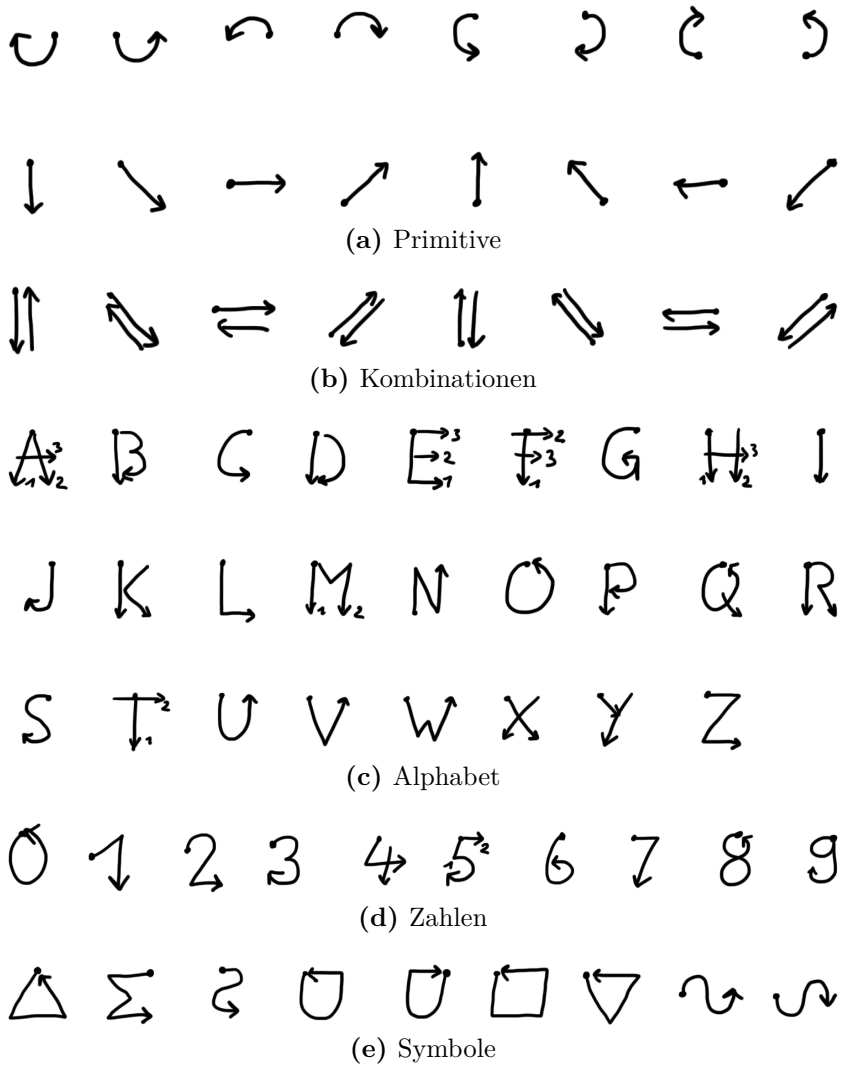
wünschenswert und wird als störend empfunden. Durch das Aufeinanderschreiben der Buchstaben wird die Notwendigkeit einer Ganzkörperbewegung vermieden. Die Probanden wurden angewiesen, das Handgelenk nicht zu bewegen und die Schreibbewegung aus dem Arm heraus auszuführen. Dies fiel einigen Probanden schwer und wurde nicht immer vollständig umgesetzt, in den Daten allerdings als natürlich auftretende Variation belassen. Konkrete Schreibfehler wurden, soweit möglich, direkt verbessert. Die Probanden konnten von ihnen bemerkte Fehlausführungen durch Wiederholung korrigieren. Da ein Schreiber im Regelfall eine Fehlausführung, also einen Schreibfehler, zuverlässig bemerkt, kann davon ausgegangen werden, dass der Datensatz nur wenige Schreibfehler enthält. Pausen konnten die Probanden nach Belieben selbständig einlegen, um eine zu starke Ermüdung oder Verkrampfung zu verhindern. Bei der Aufnahme der Wörter und Sätze wurden die Probanden explizit instruiert, dass keine Pausen zwischen Buchstaben und Wörtern notwendig sind, sondern kontinuierlich geschrieben werden soll.

3.2.3 Der Strich-Datensatz

Der Datensatz Strich wurde mit dem Handschuh aufgenommen. Der Datensatz dient als Referenzdatensatz für die Strichmodellierung, welche in Kapitel 6 beschrieben wird. Daraus ergeben sich drei wesentliche Anforderungen, die sich von den Anforderungen an die bisherigen Datensätze unterscheiden:

- Erfassung der 3D-Trajektorie: Zusätzlich zu den Inertialsensordaten soll synchron die Trajektorie im Raum erfasst werden. Anhand der Trajektorie können die Inertialsensordaten anschließend in einzelne Teilstriche zerlegt werden.
- Striche in unterschiedlichen Kontexten: Die einzelnen Striche sollten möglichst mit vielen unterschiedlichen Vor- und Nachfolgestrichen vorkommen.
- Erweiterung der Domäne: Das Gestenalphabet sollte eine größere Menge an Zeichen beinhalten. Damit kann untersucht werden, ob sich Strichmodelle, die auf einer Domäne von Gesten trainiert wurden, zur Erkennung in einer anderen Gestendomäne eignen.

Für den Datensatz wurden Daten von einem Probanden aufgenommen. Das Aufnahmeverfahren entsprach dem des Handschuh-Datensatzes, der Proband saß vor einem Rechner, führte die Gesten vor sich aus und segmentierte sie eigenständig über einen Tastendruck. Zusätzlich wurde eine Tiefenkamera



(a) Primitive

(b) Kombinationen

(c) Alphabet

(d) Zahlen

(e) Symbole

Abbildung 3.3 – Im Strich-Datensatz aufgenommene Zeichen und Bewegungen, bestehend aus den Untermengen: Primitive Bewegungen (a), Kombinationen (b), Buchstaben des Alphabets (c), Zahlen (d) und Symbole (e). Die Pfeile geben jeweils die Richtungen zusammen mit den Endpunkten einer Bewegung an. Die Zahlen geben die Reihenfolge der Ausführung an, falls Zeichen aus mehreren Bewegungen bestehen, die auf dem Papier mit Stift An- und Absetzbewegungen verbunden wären. Die Nummerierung ist nur für Zeichen angegeben, für die es keine etablierte Schreibreihenfolge gibt.

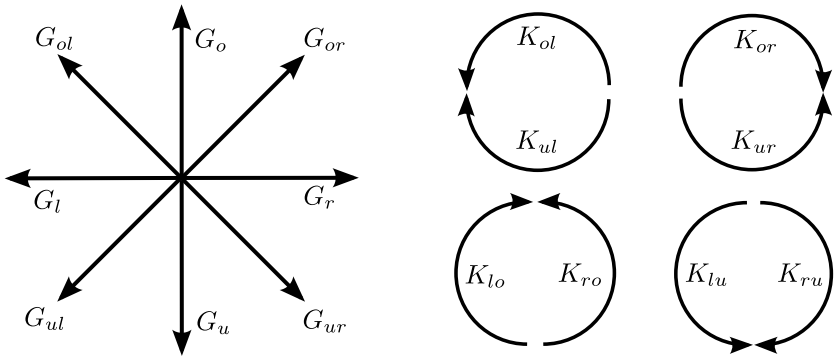


Abbildung 3.4 – Die zur Annotation verwendete Menge an Strichen. Die Kurzformen der Bezeichner der einzelnen Strichen setzen sich aus Art und Richtung der Bewegung zusammen. Dabei steht G für Gerade, K für Kreis und o, u, r, l für die Richtungen: (nach) *oben*, *unten*, *rechts*, *links*.

für die Erfassung der Trajektorie verwendet. Hierfür wurde ein *leap* Sensor der gleichnamigen Firma verwendet, der dazu gedacht ist, Hand- und Fingerbewegungen vor einem Rechner zu erfassen, um beispielsweise eine Gestensteuerung von Desktopanwendungen zu ermöglichen. Er wird auf dem Tisch vor dem Monitor platziert und eignet sich daher sehr gut, um vor dem Monitor ausgeführte Schriftbewegungen zu erfassen. Das Aufnahmevolumen in 30 cm Höhe über dem Sensor ist ausreichend groß für die Erfassung der Buchstaben und Gesten. Die zugehörige Software liefert die Trajektorie eines ausgestreckten Fingers, so dass zur Erfassung während der Durchführung der Aufnahmen nur der Zeigefinger ausgestreckt werden musste. Durch die Synchronisierung der zwei Datenströme ist es möglich, die Daten der Inertialsensoren mit den Ortsdaten zusammenzuführen und so eine Referenztrajektorie zu den Inertialsensordaten herzustellen.

Es wurden von einem Probanden 15 Wiederholungen der in Abbildung 3.3 dargestellten Zeichen aufgenommen. Abbildung 3.3 zeigt die zusätzlichen Bewegungen. Insgesamt wurden 69 unterschiedliche Zeichen aufgenommen.

Annotation

Anhand der aufgezeichneten Trajektorie wurden alle Zeichen und Bewegungen im Datensatz in einzelne primitive Striche segmentiert. Als Menge der primitiven Striche wurden acht gerade Striche und acht halbe Kreissegmente festgelegt. Abbildung 3.4 zeigt diese 16 Strichbewegungen und führt Bezeichner für sie ein. Die Wahl der Strichmenge wird in Kapitel 6.1.1 begründet.

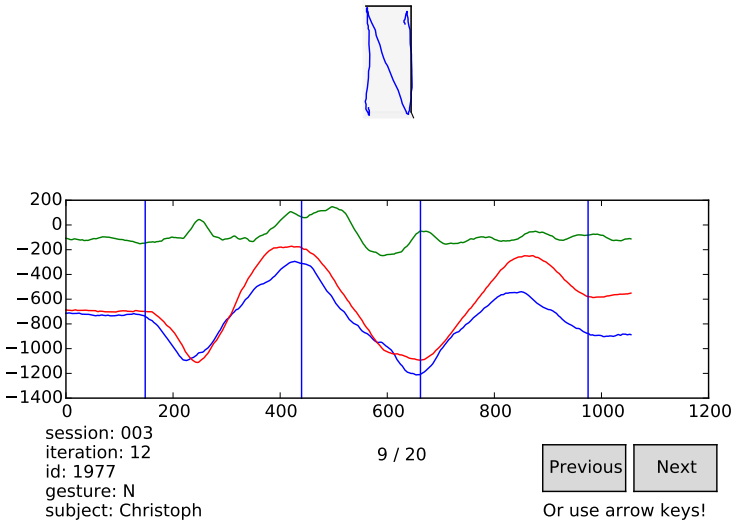


Abbildung 3.5 – Bildschirmfoto des Annotierungsprogramms zur Segmentierung der Daten in einzelne Striche am Beispiel des Buchstabens N. Im oberen Teil ist die mittels Tiefenkamera erfasste Trajektorie zu sehen. Im unteren Teil sind die Beschleunigungssignale in den drei Raumachsen in Milli-g angegeben. Die grüne Kurve entspricht weitgehend der horizontalen Beschleunigung, die vertikale Beschleunigung teilt sich auf die rote und blaue Kurve auf. Die vertikalen blauen Striche geben die Segmentierung des Buchstabens in seine drei Einzelstriche an. Wird der Mauszeiger innerhalb des Koordinatensystems mit den Beschleunigungskurven bewegt, wird in der Darstellung der Trajektorie eine Markierung angezeigt, welche die Position zeit-synchron zu den Beschleunigungsdaten anzeigt. Durch Klicken können Segmentgrenzen einfach gesetzt oder gelöscht werden.

Zusätzlich zu den Strichen wurden auch Ruhephasen vor und nach der Bewegung annotiert. Die Annotation erfolgte mit einer Software die Trajektorien- und Inertialsensordaten gleichzeitig dargestellt. Mit der Maus kann über die Inertialsensordaten gefahren werden. Ein Kreuz stellt die entsprechende Position in der Trajektorie dar. Per Mausklick können Grenzen gesetzt und gelöscht werden. Der Nutzer legt nur die Strichgrenzen fest, eine Zuordnung zu den Strichen findet später statt. Durch die Wahl der Strichmenge lassen sich naturgemäß nicht alle Bewegungen exakt auf die definierten Striche abbilden. Die Auswahl der Striche erfolgte gemäß der in Abbildung 3.3 definierten Reihenfolge der Striche und der in Anhang A.1 festgelegten Sequenzen

von Strichen für die einzelnen Zeichen. Abbildung 3.5 zeigt ein Bildschirmfoto der Annotation.

3.2.4 Der Detektion-Datensatz

Um die Qualität der Schriftdetektion in kontinuierlichen Daten zu untersuchen, wurden Daten von drei Probanden aufgenommen. Für die Detektion ist weniger die interpersonelle Varianz der Daten relevant als vielmehr eine möglichst hohe Variabilität der aufgenommenen Aktivitäten, da die Herausforderung hier in der Abgrenzung von Schrift gegenüber allen Arten anderer Aktivitäten liegt. Aus diesem Grund wurden nur drei Probanden aufgenommen, diese steuerten allerdings längere Aufnahmen während Alltagsaktivitäten in realistischem Umfeld bei. Der Datensatz wurde von den Probanden in ihrer privaten Wohnung aufgenommen. Während der Aufnahmen verrichteten die Probanden Aktivitäten, wie beispielsweise Kochen, Essen oder Wäsche aufhängen.

Der Datensatz besteht aus einem Trainingsteil und einem Evaluationsteil. Der Trainingsteil enthält ausschließlich Alltagsaktivitäten und keine Handschrift und hat eine Länge von 111 min für alle Probanden zusammen. Der Evaluationsteil besteht aus Alltagsaktivitäten mit sporadischer Handschrift. Er hat eine Länge von 115 min von denen 111 min auf Alltagsaktivitäten und 4 min auf Handschrift entfallen. In diesen vier Minuten wurden 17 Sätze mit insgesamt 68 Wörtern geschrieben. Die Probanden annotierten kurz nach dem Schreiben die Uhrzeit und den geschriebenen Text mit einem Stift auf Papier. Da es sich dabei um kleinräumige, herkömmliche Schrift handelt, führt die Annotation nicht zu weiteren, ungewollten Schriftsegmenten im Datensatz. Aufgrund des zeitlichen Versatzes stellt dies nur eine grobe Annotation der Daten dar. Über ein nachträgliches Viterbi Alignment des annotierten Handschriftsegments mit dem notierten Referenzsatz wurde die Annotation zeitlich präzisiert.

3.2.5 Der Armband-Datensatz

Der Datensatz Armband wurde mit dem Armband aufgenommen. Anhand des Datensatzes wird evaluiert, ob eine Erkennung auch möglich ist, wenn der Sensor am Handgelenk befestigt ist. Außerdem wird überprüft, ob die Möglichkeit besteht, bereits auf dem Handschuh-Datensatz trainierte Modelle zur Erkennung am Handgelenk erfasster Schriftdaten zu benutzen.

Im Gegensatz zum Handschuh-Datensatz wird den Probanden in diesem Datensatz eine freie und damit angenehmere Armhaltung erlaubt. Die Aufnahmen für den Handschuh-Datensatz führten bei den meisten Probanden bereits nach kurzer Zeit zu einer Ermüdung und Verkrampfung des Armes. Dieser auch als „Gorilla Arm“ bekannte Effekt ist von der Bedienung großflächiger Bildschirme mittels (Touch-)Gesten weithin bekannt. Zusätzlich fühlt sich das Schreiben vor dem Gesicht mit Buchstaben von ca. 20 cm Höhe seltsam für den Nutzer an und ist vermutlich auch sozial kaum akzeptabel. Dies wurde auch durch die Aussagen der Probanden bestätigt. Daher durften die Probanden für den Armband-Datensatz die Armhaltung frei wählen, waren also insbesondere nicht gezwungen vor dem Gesicht zu schreiben. Acht Probanden schrieben dieselben 80 Sätze wie im Handschuh.S Datensatz und es wurde die gleiche Software zur Steuerung der Aufnahmen verwendet.

3.3 Vergleich und Analyse der Datensätze

Durch die Unterschiede der Sensoren, der Sensorposition sowie der Art des Schreibens ergeben sich zwischen den Datensätzen Unterschiede in den Sensorsignalen. Anhand exemplarischer Beispiele sollen diese hier verdeutlicht werden. Abbildung 3.6 zeigt typische Beschleunigungs- und Winkelgeschwindigkeitsverläufe für den Buchstaben P zusammen mit der Trajektorie aus den Datensätzen Strich und Stift. Für die Signale aus dem Stift-Datensatz ist zusätzlich der Verlauf des Drucksensors angegeben. Es ist deutlich zu sehen, dass die Varianz des Beschleunigungssignals bei den Stiftdaten (Stift) geringer ist als bei den *Airwriting* (Strich) Daten. Die Trajektorie verdeutlicht das Fehlen der Stift An- und Absetzbewegung im *Airwriting* Fall. Bei den Aufnahmen mit dem Stift liefert der Drucksensor die Information, wann tatsächlich geschrieben wurde.

3.3.1 Normalisierung der Sensorposition

Ein weiterer Unterschied zwischen den Datensätzen Handschuh und Armband entsteht durch die unterschiedliche Position des Sensors. Diese führt zu unterschiedlichen Signalen bei gleicher Bewegung. Da alle Airwriting Aufnahmen im Wesentlichen ohne Bewegung des Handgelenks ausgeführt wurden, messen der Handschuh und das Armband zwar die gleiche Bewegung, aber die unterschiedliche Sensorposition führt zu unterschiedlichen Sensorkoordinatensystemen. Diese können allerdings durch Rotation und Skalierung in-

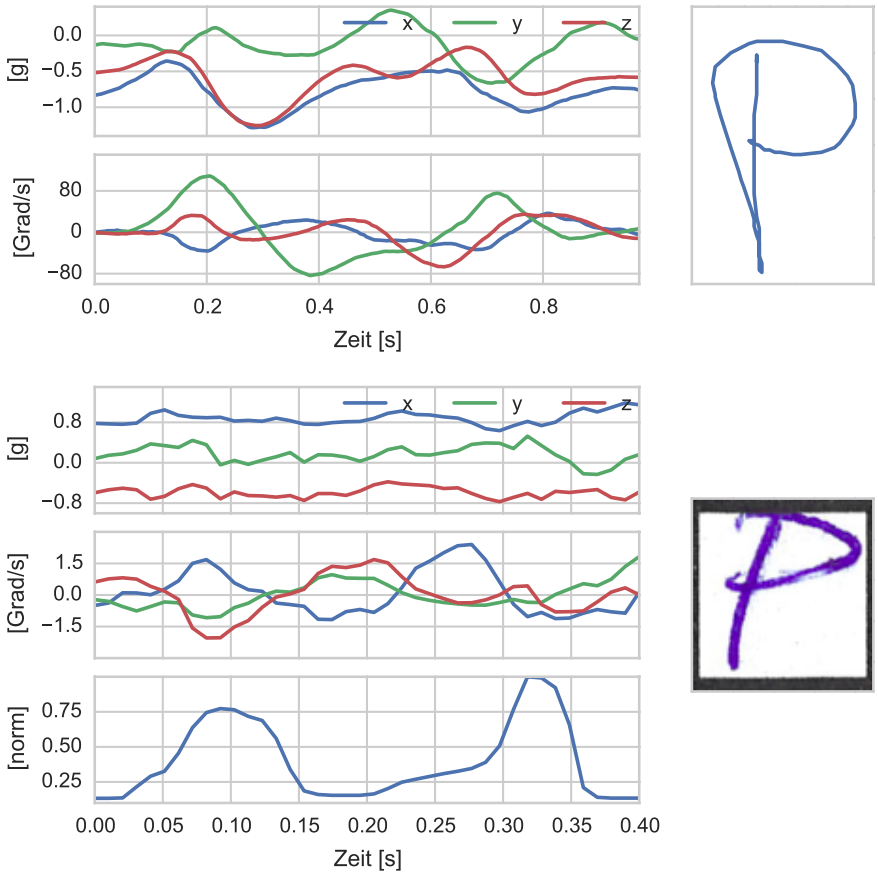


Abbildung 3.6 – Gegenüberstellung exemplarischer Signale des Handschuhs (Datensatz Strich) oben und des Sensorstifts (Datensatz Stift) unten mit zugehörigen Rohsignalen der Beschleunigungen und Winkelgeschwindigkeiten und im Falle des Stiftes auch des Drucksensors. Auf der rechten Seite sind jeweils die Trajektorien zu den Signalen dargestellt. Für den Handschuh wurde diese mit einer Tiefenkamera erfasst, für den Stift ist das Schriftbild abgebildet.

einander überführt werden. Um beide Datensätze anzugleichen, werden die Daten des Armbandsensors in das Koordinatensystem des Handschuhsensors transformiert. Hierfür wurde eine parallele Aufnahme einer Bewegung mit beiden Sensoren durchgeführt. Die Daten des Armbandes wurden durch eine Abtastratenerhöhung auf die gleiche Abtastrate gebracht wie die Handschuhdaten. Danach wurde von jedem Sensorkanal der Mittelwert subtrahiert, um den Einfluss der Gravitation weitgehend auszuschalten. Anschließend wurde diejenige lineare Transformation bestimmt, welche den quadratischen Abstand zwischen den beiden Signalen minimiert. Die Transformation besteht dabei aus einer orthogonalen Rotation und einer Skalierung. Seien die Signale als Matrizen X und Y gegeben, dann wird diejenige Transformierte Z unter der Transformation

$$Z = b \cdot Y \cdot C$$

gesucht, für welche die Summe der quadratischen Abweichung von Z und X minimal ist. Die Transformation ist durch den Skalierungsfaktor b und die orthogonale Rotationsmatrix C gegeben. Das lineare Gleichungssystem ist eindeutig lösbar. Durch die Transformation der Daten wurde die Summe der quadratischen Fehler um 57% gegenüber dem Fehler zwischen den Originalsignalen reduziert. Die berechnete Transformation wurde für die Normalisierung des Datensatzes Armband verwendet.

3.3.2 Personenspezifische Varianz

Wie bereits in Abschnitt 2.1 beschrieben, gibt es große Unterschiede im Schriftbild zwischen unterschiedlichen Schreibern. Die Unterschiede beziehen sich nicht nur auf die graphische Repräsentation sondern auch auf die Abfolge der Striche, aus denen ein Buchstabe zusammengesetzt ist. Für den Datensatz Handschuh.B wurden die unterschiedlichen Schreibvarianten während der Aufnahme annotiert. Varianten eines Buchstabens sind durch eine sich unterscheidende Folge von Strichen definiert. Tabelle 3.2 gibt einen Überblick über die Anzahl der beobachteten Schreibvarianten der Buchstaben im Datensatz Handschuh.B an. Da der Datensatz Handschuh nur Daten von 9 Probanden enthält, liefert die Tabelle allerdings nur eine untere Abschätzung der tatsächlich geläufigen Schreibvarianten. Im Anhang A.1 werden die einzelnen Varianten mit der zugehörigen Strichreihenfolge aufgelistet.

Buchstabe	A	B	C	D	E	F	G	H	I	J	K	L	M
Anzahl Varianten	3	1	1	1	5	3	5	2	2	6	2	1	2

Buchstabe	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Anzahl Varianten	2	1	1	1	1	1	2	3	1	1	3	3	2

Tabelle 3.2 – Anzahl der beobachteten Schreibvarianten für große Druckbuchstaben im Datensatz Handschuh.B.

3.4 Zusammenfassung

In diesem Kapitel wurden alle in dieser Arbeit verwendeten Datensätze mit Schriftdaten vorgestellt. Dies beinhaltet sowohl die verwendete Sensorik als auch eine genaue Beschreibung der Aufnahmebedingungen und des verwendeten Aufnahmeprotokolls. Für den Datensatz Strich wird zudem eine Menge von primitiven Strichen eingeführt, aus denen die einzelnen enthaltenen Zeichen zusammengesetzt sind. Dies wird in Kapitel 6 als Grundlage für eine strichbasierte Modellierung dienen. Die Stift- und Airwriting-Daten der Datensätze Stift und Handschuh werden gegenübergestellt. Es wird eine Methode beschrieben, um die mit dem Armband aufgenommenen Daten, in das Koordinatensystem des Handschuhs zu transformieren. Dies ermöglicht beispielsweise die Verwendung von Modellen, die auf Handschuhdaten trainiert wurden, für die Erkennung von Schrift, die mit dem Armband erfasst wird.

Kapitel 4

Automatische Schrifterkennung

In diesem Kapitel wird die Architektur des Systems zur Schrifterkennung beschrieben. Die einzelnen Komponenten werden eingeführt und ihre Funktionsweisen erläutert. Die grundlegende Methodik zur Initialisierung, zum Training und zur Dekodierung wird ebenfalls vorgestellt.

4.1 Überblick

Abbildung 4.1 gibt die Struktur des Handschrifterkennungssystems mit allen relevanten Komponenten wieder. Im Wesentlichen entspricht der Aufbau einem klassischen Spracherkennungssystem, da das zu lösende Trainings- und Erkennungsproblem vergleichbar ist: Für ein gegebenes kontinuierliches Eingangssignal soll die wahrscheinlichste darin kodierte Wortfolge gefunden werden. Statt eines akustischen Signals wird im vorliegenden Fall ein Bewegungssignal verarbeitet. Aus dem Rohsignal wird eine Sequenz von Merkmalsvektoren generiert. Ein Wörterbuch spezifiziert den Wortschatz des Erkennungssystems und gibt für jedes Wort an, aus welcher Sequenz von atomaren Einheiten es zusammengesetzt ist. In der Sprache sind die atomaren Einheiten in der Regel Phoneme, im Falle von Handschrift häufig Buchstaben. Die Muster der Buchstabensequenzen im Merkmalsraum werden statistisch modelliert. Das statistische Modell der Phoneme als atomare Einheiten wird in der Spracherkennung als „akustisches Modell“ bezeichnet. Analog wird

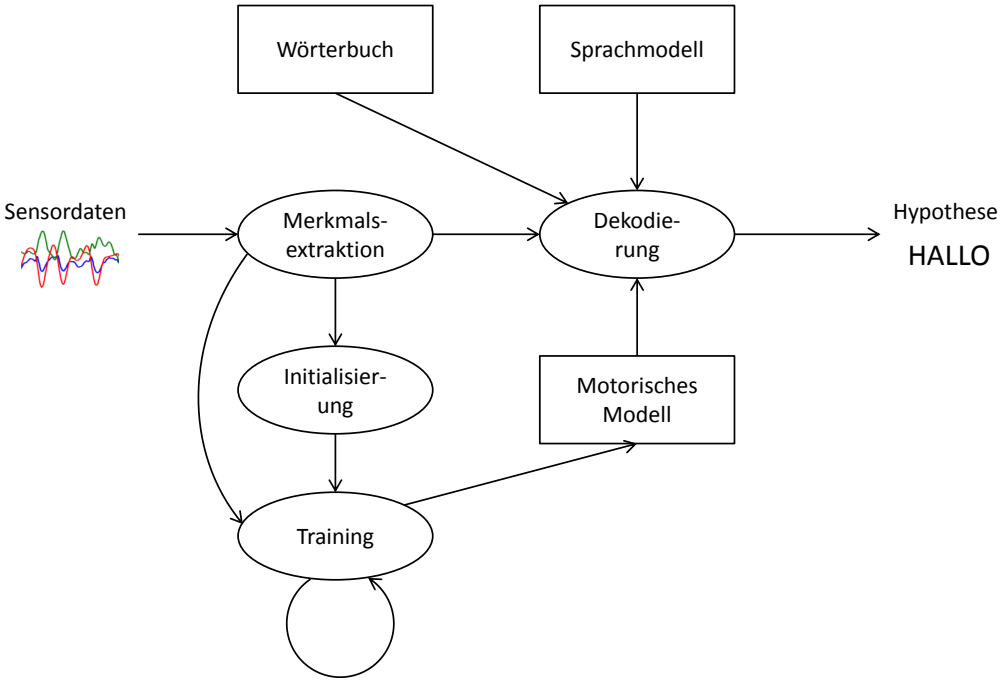


Abbildung 4.1 – Struktur eines HMM Handschrifterkennungssystems.

in dieser Arbeit vom „motorischen Modell“ für die atomaren Repräsentationen der Schreibbewegungen gesprochen. Ein Sprachmodell liefert zusätzlich Informationen über die Struktur der Zielsprache, indem es die Auftrittswahrscheinlichkeiten von Wortfolgen modelliert. Damit wird auch ein Bezug zur Semantik hergestellt, da beispielsweise unsinnige Wortfolgen eine geringe Wahrscheinlichkeit aufweisen.

Die Algorithmen für die Initialisierung und das Training der motorischen Modelle sowie die effiziente Dekodierung sind aus der Spracherkennung übertragbar. In den nachfolgenden Abschnitten werden die einzelnen Komponenten näher erläutert. Die Implementierung des Basissystems und der darauf aufbauenden Erweiterungen wurde teilweise mit dem Janus Recognition Toolkit [SMFW01] und teilweise mit dem Biosignals Toolkit (BioKIT) [TWG+14] realisiert.

4.2 Merkmalsextraktion

Das Ziel der Merkmalsextraktion ist die Extraktion der für die Mustererkennungsaufgabe relevanten Information aus den Rohdaten. In diesem Sinne nicht relevante Informationen sollen nach Möglichkeit in den extrahierten Merkmalen nicht mehr vorhanden sein. Dadurch erhöht sich einerseits die Generalisierungsfähigkeit des Erkenners und durch eine Reduktion des Datenvolumens müssen weniger Parameter der statistischen Modelle geschätzt werden. Für den vorliegenden Fall soll die Merkmalsextraktion den Einfluss der Erdbeschleunigung sowie der Schreibgeschwindigkeit reduzieren. Wie in Absatz 2.2 beschrieben, wird die Erdbeschleunigung mit einem konstanten Wert von 1 g permanent gemessen. Dabei wird diese Beschleunigung anteilig in allen drei Achsen des Sensors abhängig von dessen Orientierung im Raum gemessen. Die Orientierung des Sensors ist nicht bekannt und damit kann die Erdbeschleunigung nicht von den gemessenen Daten abgezogen werden. Dies ist nur möglich, wenn der Sensor keine weitere Beschleunigung erfährt. Bleibt die Orientierung allerdings während einer Bewegung konstant, so führt die Erdbeschleunigung zu einem konstanten Versatz in den Daten und kann über eine Mittelwertbereinigung neutralisiert werden. Unterschiede in der Schreibgeschwindigkeit führen, wie in Abschnitt 2.1 erwähnt, zu Unterschieden in der Amplitude der Beschleunigungs- und Drehratendaten. Diese werden durch eine Varianznormalisierung ausgeglichen. Über eine Fensterung und Mittelwertbildung werden die Daten zusätzlich geglättet und auf eine einheitliche Merkmalsrate von 100 Hz gebracht. Dies ist die maximale Abtastfrequenz über alle in dieser Arbeit verwendeten Sensoren und sie ist zur Erfassung menschlicher Bewegungen ausreichend hoch. Soweit in den nachfolgenden Kapiteln nicht anders angegeben, werden die Daten nach den folgenden Schritten vorverarbeitet:

1. Fensterung: Fensterung der Eingangsdaten mit einer rechteckigen Fensterfunktion und einer Fensterbreite von 10 ms ohne Fensterüberlappung. Pro Fenster wird der Mittelwert berechnet.

2. Z-Normalisierung: Mittelwert und Varianz werden normalisiert, indem pro Kanal der Mittelwert subtrahiert wird und durch die Standardabweichung geteilt wird.

Da keine Merkmale im Frequenzbereich berechnet werden, ist eine rechteckige Fensterfunktion problemlos anwendbar. Überlappende Fenster bringen keine Vorteile, da dieser Schritt der Merkmalsextraktion darauf abzielt, ein geglättetes Signal mit einheitlicher Merkmalsrate zu berechnen. In informellen Experimenten wurde dies bestätigt.

4.3 Motorisches Modell

Das motorische Modell beinhaltet die HMMs aller atomaren Modellierungseinheiten. Bezogen auf die vorliegende Arbeit sind dies entweder Buchstaben oder einzelne Striche. Im Falle von Buchstaben liegt beispielsweise entsprechend für jeden Buchstaben ein HMM vor, welches den typischen Verlauf der Merkmale für den jeweiligen Buchstaben beschreibt.

4.4 Wörterbuch

Das Wörterbuch definiert die tatsächlich erkennbaren Einheiten und ordnet ihnen eine Folge von atomaren Modellierungseinheiten zu, aus denen sie zusammengesetzt werden. Im Falle einer Modellierung von Schrift mit Buchstaben als atomare Einheiten, stehen im Wörterbuch alle erkennbaren Wörter und die Sequenz von Buchstaben, aus denen sie gebildet werden. Die Zuordnung ist in diesem Fall trivial und ergibt sich direkt aus der Orthographie. Im Falle der in Kapitel 6 beschriebenen Modellierung mit Strichen als atomare Einheiten, ist die Zuordnung nicht mehr trivial und auch nicht eindeutig. So kann ein großes U beispielsweise mit oder ohne einem abschließenden gerade nach unten führenden Strich geschrieben werden. Damit existieren zwei unterschiedliche Strichsequenzen für die semantisch gleiche erkennbare Einheit eines Buchstabens. Im Wörterbuch können für diesen Fall Schreibvarianten einer erkennbaren Einheit angegeben werden. Dies entspricht den durch unterschiedliche Phonemsequenzen charakterisierten Aussprachevarianten von Wörtern in der Spracherkennung. Die Varianten stehen dabei gleichberechtigt nebeneinander.

Eine Erkennung von nicht im Wörterbuch vorhandenen Wörtern wird im Rahmen dieser Arbeit nicht behandelt. Für ein gegebenes Eingangssignal gibt der Erkenner entsprechend immer die wahrscheinlichste Sequenz von Wörtern aus dem Wörterbuch aus. Mögliche Ansatzpunkte, um in Zukunft auch unbekannte Wörter erkennen zu können, wären eine freie Erkennung als Buchstabensequenz.

4.5 Sprachmodell

Ein Sprachmodell gibt die Wahrscheinlichkeit einer Wortsequenz an. In dieser Arbeit werden ausschließlich statistische N-Gramm Sprachmodelle verwen-

det, welche im Folgenden beschrieben werden. Ein N-Gramm Sprachmodell liefert für ein Wort w_t, gegeben die Historie der Wörter $w_{t-1}, \ldots, w_{t-N+1}$, die bedingte Wahrscheinlichkeit

$$P(w_t|w_{t-1}, \ldots, w_{t-N+1}) \,.$$

Für den Fall $N = 0$ ist die Wahrscheinlichkeitsverteilung über alle Wörter als Gleichverteilung modelliert. Die Wahrscheinlichkeiten werden in der Regel empirisch auf großen Textkorpora bestimmt. Die Qualität eines Sprachmodells auf einer Testmenge von Wortsequenzen, also beispielsweise einem Text, lässt sich durch die N-Gramm Abdeckung und die Perplexität beschreiben. Die N-Gramm Abdeckung gibt prozentual an, wieviele der im Testtext vorgefundenen N-Gramme im Sprachmodell vorkommen. Kommt ein N-Gramm nicht im Modell vor, wird auf das (N-1)-Gramm zurückgegriffen. Für gänzlich unbekannte Wörter wird auf einen hierfür definierten Wert zurückgegriffen, beispielsweise auf eine Rückfall-Wahrscheinlichkeit (engl. backoff). Für die Experimente in dieser Arbeit kommt dieser Fall aber nicht vor. Die Perplexität ist ein informationstheoretisches Maß und liefert eine Aussage über die Prädiktionsqualität des Modells auf der Testmenge. Die Perplexität kann als die mittlere Anzahl von Möglichkeiten für das nächste Wort interpretiert werden, falls das Modell gleichverteilt und unabhängig zwischen allen Möglichkeiten wählen würde. Im Falle einer tatsächlichen Gleichverteilung, also einem 0-Gramm Sprachmodell, entspricht die Perplexität genau der Anzahl der Wörter im Wörterbuch.

4.6 Initialisierung und Training

Die Initialisierung erfolgt gemäß des in Abschnitt 2.3.2 beschriebenen Verfahrens. Um eine möglichst gute initiale Schätzung der Zuordnung von Signalbereichen zu den HMM-Zuständen zu erhalten, wird die Initialisierung ausschließlich auf atomaren Modellen durchgeführt. Im Falle von Buchstaben werden also isolierte Buchstabenaufnahmen, im Falle von Strichen isolierte Strichaufnahmen oder manuell annotierte Aufnahmen verwendet.

Das Training erfolgt durch ein mehrfaches Trainieren auf sukzessive komplexeren Daten. Komplex bedeutet in diesem Falle länger im Sinne der Anzahl von atomaren Einheiten. Die Daten werden beispielsweise auf Buchstabenaufnahmen initialiert und trainiert, anschließend erfolgt ein weiteres Training der Modelle auf Wortaufnahmen und abschließend auf Aufnahmen ganzer Sätze.

4.7 Dekodierung

In Abschnitt 2.3.3 wurden der Forward und der Viterbi Algorithmus als mögliche Verfahren für eine Klassifikation mit HMMs benannt. Bei einer kontinuierlichen Erkennung von Handschrift ist der Suchraum für eine effiziente Berechnung mit diesen Verfahren in der Regel zu groß, da das Wörterbuch mehrere Tausend Wörter umfasst und für eine gegebene Signalsequenz eine noch viel höhere Anzahl an Wortsequenzen möglich ist. Daher verwendet man beispielsweise eine Viterbi-Beam-Suche. Das Verfahren wurde ursprünglich für die Spracherkennung entwickelt.

Dazu wird für das gesamte Wörterbuch wird ein Suchgraph aufgebaut. Das Wörterbuch gibt dabei die Menge der erkennbaren Einheiten sowie deren Zusammensetztung aus atomaren HMMs an. Um identische Wortanfänge nur einmal evaluieren zu müssen, wird der Suchgraph in Form eines Präfixbaumes erstellt [HAH01]. Die Suche erfolgt zeitlich sequentiell, es wird also Merkmalsvektor nach Merkmalsvektor verarbeitet, indem Hypothesen gemäß des Viterbi Kriteriums durch den Suchgraph propagiert werden. Jede Hypothese entspricht dabei einem möglichen Pfad. Wenn weder Anzahl noch Wahrscheinlichkeiten der Hypothesen beschränkt werden, entspricht dies dem vollständigen Viterbi Algorithmus. Über mehrere *Pruning* Parameter kann sowohl die Anzahl der für jeden Zeitschritt aktiven Hypothesen als auch die maximale Differenz der Wahrscheinlichkeit zur aktuell besten Hypothese (Beam) beschränkt werden. Damit lässt sich der Rechenaufwand gegen die Erkennungsgenauigkeit einstellen und eine effiziente Berechnung auf Kosten der Genauigkeit wird möglich [OVWY94]. Nach Erreichen eines Endzustandes eines Wortes wird die Hypothese wieder in den Anfangszustand des Suchgraphen überführt. Das Sprachmodell und das motorische Modell werden dabei in Anlehnung an die Bayes Formel (Gleichung 4.1) miteinander kombiniert. Sei mit O die Beobachtung, also die Sequenz an Merkmalsvektoren, gegeben und bezeichne W die Menge aller möglichen Wortsequenzen. Dann wird die Wortsequenz w mit maximaler Wahrscheinlichkeit gemäß

$$\arg\max_{w \in W} P(w|O) = \arg\max_{w \in W} \frac{P(O|w)P(w)}{P(O)} \tag{4.1}$$

gesucht. Dabei entspricht die Wahrscheinlichkeit $P(O|w)$ dem motorischen Modell und $P(w)$ dem Sprachmodell. Die Wahrscheinlichkeit $P(O)$ ist unabhängig von w und daher für die Berechnung irrelevant. Zusätzlich werden über einen Faktor das Sprachmodell gegenüber dem motorischen Modell gewichtet. Ein weiterer Parameter gibt einen Straffaktor an, der die Wahrscheinlichkeit einer Wortsequenz linear mit der Anzahl der erkannten Wörter

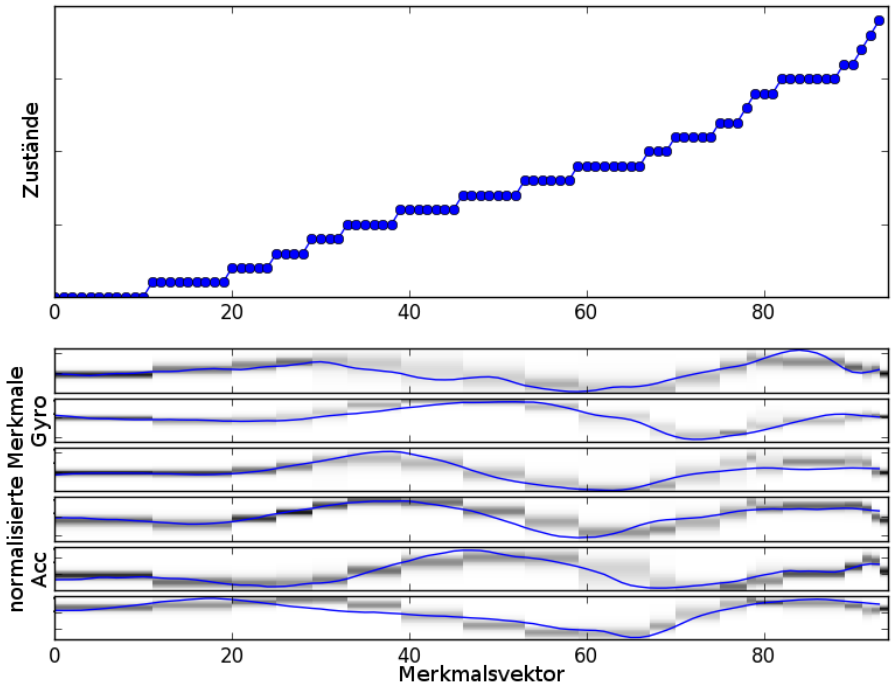


Abbildung 4.2 – Viterbi Pfad für den Buchstaben O. Die obere Grafik zeigt, in welchem Zustand sich das Modell zu welchem Zeitpunkt befindet. Die untere Grafik zeigt die Sequenz der Merkmalsvektoren, welche im Wesentlichen dem Signalverlauf entspricht, als blaue Linie. Die unterlegten Graustufenbilder zeigen die Höhe der Wahrscheinlichkeitsdichte des zum jeweiligen Zustand gehörenden Modells.

senkt. Dies soll die Erkennung sehr vieler sehr kurzer Wörter verhindern. Die konkreten Werte dieser Parameter werden in der Regel empirisch bestimmt.

Nachdem der letzte Merkmalsvektor verarbeitet ist, liefert die Hypothese mit der höchsten Wahrscheinlichkeit das Erkennungsergebnis und der Pfad dieser Hypothese kann zurückverfolgt werden. Abbildung 4.2 zeigt einen Airwriting Pfad für den Buchstaben O mitsamt der Markmalsvektorsequenz und den Mixturmodellen.

Kapitel 5

Buchstabenbasierte Erkennung

In diesem Kapitel wird eine Modellierung auf Buchstabenebene vorgestellt. Der Ansatz wird in mehreren Experimenten für den Sensorstift und den Airwriting Fall evaluiert. Dabei werden Ergebnisse für die Erkennung einzelner Buchstaben, ganzer Wörter und kontinuierlich geschriebener Sätze präsentiert. Die vorgestellten Ergebnisse zeigen erstmalig, dass eine Texterkennung mit Inertialsensorik sowohl für den Sensorstift als auch für Airwriting möglich ist und praktikable Fehlerraten erzielt werden können.

5.1 Buchstaben als atomare Einheiten

Für Schrift, die auf dem lateinischen Alphabet beruht, ist es naheliegend, Buchstaben als atomare Modellierungseinheiten zu verwenden. Für jeden Buchstaben wird also ein HMM erstellt, welches den typischen Signalverlauf dieses Buchstabens modelliert. Ist für jeden Buchstaben ein Modell vorhanden, können beliebige Wortmodelle aus den Buchstabenmodellen erstellt werden. Jeder Buchstabe hat bis auf einige Varianten eine definierte graphische Form. Durch die Verwendung von Druckschrift wird die Anzahl der Varianten eingeschränkt. Allerdings gibt es, wie in Abschnitt 3.3.2 beschrieben, auch Unterschiede in der Schreibbewegung, um die gleiche graphische Form zu erzeugen. Die Unterschiede werden durch komplexere Verteilungen

der Beobachtungswahrscheinlichkeiten modelliert, also durch eine genügend hohe Anzahl von Komponenten in den Gaußschen Mischverteilungen.

5.2 Eingabemodalitäten

Die in diesem Kapitel vorgestellte Modellierung wird experimentell sowohl für die Anwendung mit dem Sensorstift als auch der Anwendung mit Airwriting evaluiert. Aufgrund der Gemeinsamkeiten zwischen den beiden Anwendungen wird das in Kapitel 4 eingeführte Erkennungssystem für beide als Basis verwendet. Aufbauend auf dem Basissystem, werden in diesem Kapitel an die jeweilige Anwendung angepasste Systeme beschrieben. Im Folgenden werden die Gemeinsamkeiten und Unterschiede genauer erläutert.

Die Erkennung von mit einem Stift geschriebener Handschrift ist der klassische Anwendungsfall der Online-Handschrifterkennung. Durch die Verwendung von Inertialsensoren zur Erfassung der Stiftbewegung lassen sich die in der Handschrifterkennung entwickelten Methoden allerdings nur bedingt übertragen, da auch im Falle eines Stiftes mit Inertialsensorik keine exakte Rekonstruktion der Stifttrajektorie möglich ist.

Im Gegensatz zur Erkennung von Schrift in der Luft gibt es beim Schreiben mit einem Stift die üblichen An- und Absetzbewegungen (englisch Pen-up und Pen-down). Diese können mit einem Drucksensor hinter der Mine erfasst werden und stehen zusätzlich zu den Daten der Inertialsensoren zur Verfügung. Die An- und Absetzinformation liefert für die Erkennung einzelner Buchstaben sowie für die Segmentierung von Buchstaben und Wörtern eine wichtige Information.

Die Bewegungen während des Schreibens mit einem Stift sind deutlich kleinräumiger als während des Schreibens in der Luft. Die Höhe eines Buchstabens beim Airwriting beträgt ca. 10 cm, wohingegen ein normal geschriebener Buchstabe nur eine Höhe von ca. 0,5 cm hat. Auch das Schreiben auf einer Oberfläche statt in der Luft hat Konsequenzen, insbesondere für das Beschleunigungssignal. Das Aufsetzen des Stiftes führt zu einer deutlichen Spitze in der Beschleunigung.

Im Gegensatz zum Airwriting ist der Sensor nicht mehr fix mit einem Körperteil des Nutzers verbunden, sondern kann frei in der Hand bewegt werden. Insbesondere kann der Nutzer den Stift entlang der Längsachse frei rotieren. Dadurch wird das Koordinatensystem des Stiftes rotiert, das globale Schreibkoordinatensystem bleibt aber konstant. Diese Rotation muss entwe-

der unterbunden oder detektiert und kompensiert werden. Für die durchgeführten Experimente wurden die Nutzer angehalten, den Stift nicht entlang der Längsachse zu rotieren. Eine Möglichkeit zur Kompensation bestünde in der Schätzung der Orientierung über ein geeignetes Verfahren (siehe Abschnitt 2.2) und einer anschließenden Normalisierung durch eine Rotation des Koordinatensystems auf eine festgelegte Standardorientierung.

5.3 Erweiterungen des Basissystems

Für die Erkennung von Airwriting wird das Basissystem ohne Modifikation verwendet, für die Erkennung der Sensorstift-Schrift wurden verschiedene Erweiterungen und Optimierungen vorgenommen. Im Folgenden werden die vorgenommenen Erweiterungen des in Kapitel 4 beschriebenen Basissystems erläutert.

5.3.1 Merkmalsextraktion

Die Merkmalsextraktion entspricht für die Inertialsensordaten dem in Abschnitt 4.2 beschriebenen Verfahren. Da die Abtastfrequenz der Sensoren im Stift 100 Hz beträgt und damit pro 10 ms nur ein Sensorwert erfasst wird, findet keine Fensterung und Mittelung auf dem Fenster statt. Für das Drucksignal werden zwei unterschiedliche Methoden zur Merkmalsextraktion verglichen. Dies sind:

- Methode D_{ZNorm}: Z-Normalisierung des Drucksignals analog zu den Inertialsensordaten
- Methode D_{Linear}: Normalisierung über eine lineare Skalierung des Drucksignals mit der in Abschnitt 5.3.2 beschriebenen Methode

Die normalisierten Daten werden dann jeweils als Merkmale verwendet.

5.3.2 Normalisierung des Drucksignals

Während der Aufnahmen mit dem Sensorstift stellte sich heraus, dass der Drucksensor nicht immer konsistente Werte liefert. Da der Sensor über eine Feder realisiert ist, die ihre Eigenschaften über die Nutzungsdauer ändert, kann sich der Nullpunkt des Sensors verschieben. Zusätzlich kann es unter Umständen zu einer verzögerten Rückstellung der Feder nach Wegnahme des

Drucks kommen und damit zu einem verzögerten Erreichen des Nullniveaus. Um beide Effekte zu kompensieren, wurde die im Folgenden beschriebene Vorverarbeitung entwickelt. Abbildung 5.1 illustriert die einzelnen Schritte anhand eines realen Signals.

1. Das rohe Drucksignal wird mit einem gleitenden Mittelwertfilter gefiltert, anschließend wird das Minimum des Signals abgezogen und das Signal linear auf den Wertebereich $[0, 1]$ skaliert. Über einen Schwellwert von 0.2 wird das Signal in Regionen mit anliegendem Druck und keinem Druck eingeteilt. Der Schwellwert wurde experimentell bestimmt.
2. Die Ableitung des Signals aus Schritt 1 wird auf das Intervall $[0, 1]$ skaliert und über einen Schwellwert von 0.1 werden die Regionen mit Druckanstieg oder Druckabfall identifiziert. Der Schwellwert wurde experimentell bestimmt.
3. Die identifizierten Regionen aus den Schritten 1 und 2 werden vereinigt und als Regionen mit einem Druck größer 0 definiert.
4. Jeweils zehn Messwerte am linken und rechten Rand werden als Regionen ohne Druck definiert. Vom Originalsignal wird das Minimum der Druckregionen subtrahiert. Alle Samples außerhalb der Druckregionen werden auf 0 gesetzt. Das Signal wird auf das Intervall $[0, 1]$ skaliert.

5.3.3 Segmentierung

Ziel der Segmentierung ist das Abschneiden irrelevanter Daten vor und nach dem Schreiben des Buchstabens. Hierfür wird das Drucksignal, wie in Abschnitt 5.3.2 beschrieben, normalisiert. Dann werden alle Werte am Anfang und Ende entfernt, die einen Druckwert von 0 haben.

5.4 Modellierung

5.4.1 Modellierung des Drucksensors

Der Anpressdruck der Mine wird als kontinuierliche Größe gemäß der in Abschnitt 5.3.2 beschriebenen Vorverarbeitung sequenziell modelliert. Da die Druckinformation zeitsynchron mit der Bewegung ist, werden die Inertialsensordaten auf Merkmalsebene mit dem Druckkanal fusioniert. Die Verwen-

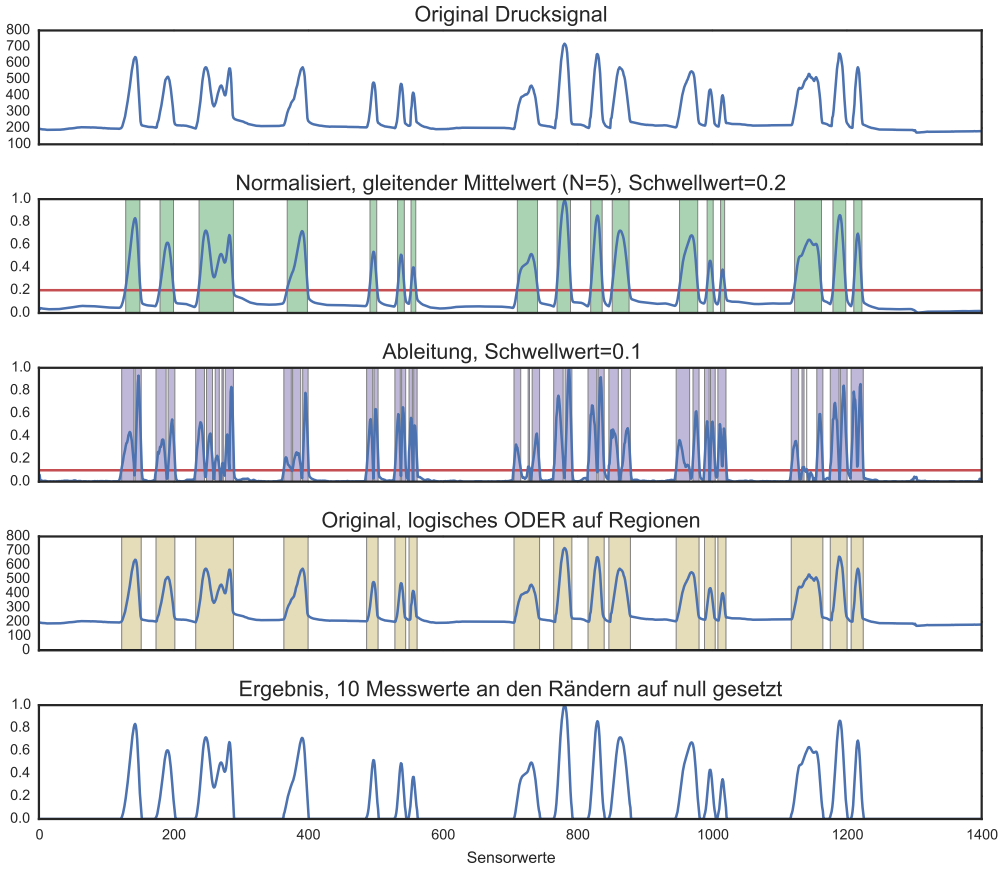


Abbildung 5.1 – Vorverarbeitungsschritte zur Normalisierung des Drucksignals am Beispiel des Wortes PRÄSIDENT. Für das Originalsignal (erste und vierte Grafik von oben) sind die rohen Sensorwerte in mV abgetragen. Für die anderen Grafiken sind die berechneten normalisierten Werte abgetragen.

dung des Drucksensors erhöht entsprechend die Dimensionalität des Merkmalsraums um eins.

5.4.2 Optimierung der HMM Topologie

Die Buchstaben des Alphabets unterscheiden sich in der Länge der Trajektorie und damit auch in der Ausführungsdauer. Eine Modellierung mit einer festen Anzahl von Zuständen muss also unterschiedlich lange Signale auf die gleiche Anzahl von Zuständen abbilden. Abbildung 5.2 zeigt die Dauer in Messwerten der Großbuchstaben aus dem Stift.G Datensatz.

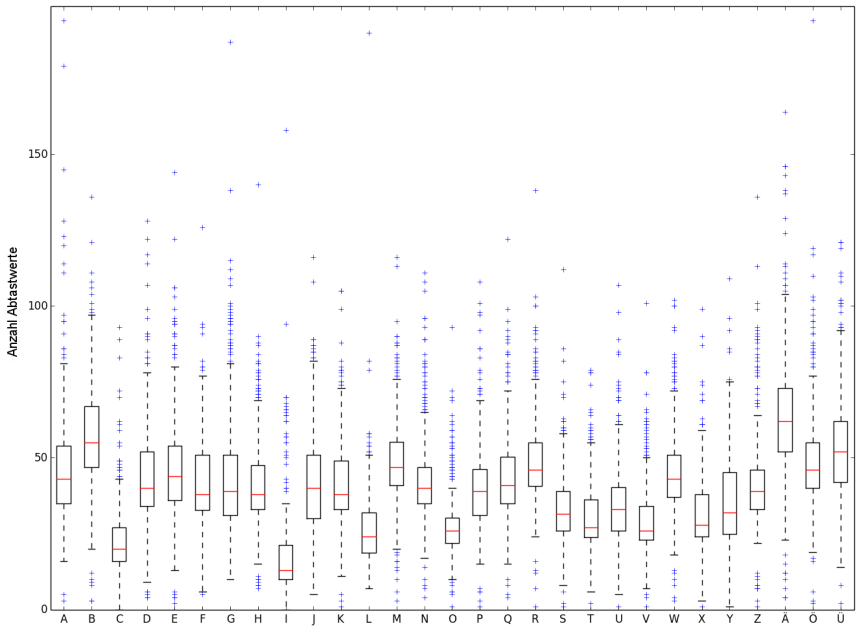


Abbildung 5.2 – Länge in Messwerten nach Großbuchstaben.

Zwischen kurzen Buchstaben wie I oder C und langen Buchstaben wie B oder Ä bestehen Unterschiede in der Länge und damit der Anzahl der Abtastwerte. Die Dauer ergibt sich aus der Komplexität des Buchstabens. Damit erhöht sich auch die Zahl der aufsteigenden und abfallende Flanken im Signal. Aus diesem Grund wird die Anzahl der Zustände pro Buchstabenmodell abhängig vom Median der zeitlichen Dauer des Buchstabens festgelegt. Relevant ist in diesem Fall aber nicht die tatsächliche Dauer, sondern die Dauer in Merkmalsvektoren. Die Topologieoptimierung ist also abhängig von dem gewählten Verfahren zur Merkmalsextraktion. Der Median wird verwendet, um den Einfluss von Ausreißern zu verringern. Ausreißer können beispielsweise durch Verweilen in der Endposition mit angesetztem Stift nach dem Schreiben entstehen oder aber auch durch eine fehlerhafte Segmentierung.

Die Gesamtzahl der Zustände wird durch die durchschnittliche Anzahl von Merkmalsvektoren pro Zustand definiert und mit τ bezeichnet. Für die Bestimmung der Dauer eines Buchstabens werden alle Merkmalsvektoren mit einem Druck größer als null gezählt. In den Regionen ohne Druck ist der Stift abgesetzt. Für jede Region mit Druck gleich null innerhalb des Buchstabens

wird ein Zusatzzustand eingefügt. Die Anzahl von Zuständen Z_B für einen Buchstaben B berechnet sich dann gemäß

$$Z_B = \frac{m_B}{\tau} + d_B \quad ,$$

wobei m_B die Anzahl der Merkmalsvektoren mit Druck größer als null und d_B die Anzahl der Regionen mit Druck gleich null innerhalb des Buchstabens angibt.

5.4.3 An- und abgesetzte Bewegung

Phasen mit abgesetztem Stift treten innerhalb eines Buchstabens wie auch zwischen Buchstaben auf. Durch die Einschränkung auf Druckschrift wird eine abgesetzte Phase zwischen Buchstaben erzwungen. *Innerhalb* einzelner Buchstaben ist die Existenz von abgesetzten Bewegungsphasen sowohl vom Buchstaben als auch von der individuellen Schreibbewegung abhängig. Manche Schreiber setzen den Stift nicht vollständig vom Blatt ab. Abgesetzte Phasen innerhalb eines Buchstabens werden implizit über die HMM Zustände des Buchstabenmodells modelliert. Durch das Training wird in einigen der Zustände des Modells die abgesetzte Bewegung modelliert. Die Datenanalyse ergab, dass typischerweise jede abgesetzte Phase durch einen Zustand modelliert wird.

Für die abgesetzte Bewegungsphase *zwischen* einzelnen Buchstaben wird ein spezielles Modell eingeführt. Die Bewegung zwischen zwei Buchstaben kann entweder stationär oder sequenziell modelliert werden. Bei einer sequenziellen Modellierung wird die Bewegung des Stiftes zwischen dem Endpunkt des vorangegangenen Buchstabens zum Startpunkt des folgenden Buchstabens modelliert. Die Analyse der aufgenommenen Daten ergab allerdings, dass Schreiber nicht immer konsistent in diesen Zwischenbewegungen sind. So werden zwischen Buchstaben manchmal Pausen gemacht, die mit einem größeren Abstand von Stiftspitze zum Papier einhergehen und damit eine ungewöhnliche große, orthogonal zur Papierebene gerichtete, Bewegung enthalten. Zusätzlich hängt die Bewegung zwischen den Buchstaben von der Position in einem Wort ab. Bei längeren Wörtern wird die Hand häufig innerhalb eines Wortes nach rechts verschoben. Dies findet in der Regel zwischen Buchstaben und nicht innerhalb eines Buchstabens statt, allerdings nicht zwischen jedem Buchstaben. Die Zwischenbewegung unterscheidet sich für den Fall, dass die ganze Hand bewegt wird, von dem Fall, dass die Hand nicht bewegt wird. Aus den genannten Gründen erscheint eine Modellierung

der Bewegung zwischen einzelnen Buchstaben wenig sinnvoll und es wurde eine stationäre Modellierung gewählt.

Bei einer stationären Modellierung besteht das Modell nur aus einem einzelnen Zustand. Die Bewegung wird also nicht in ihrem zeitlichen Verlauf modelliert. Die Idee ist hier, den Anpressdruck des Stiftes zu modellieren. Dieser ist während der abgesetzten Bewegungsphasen immer gleich null und am Endpunkt sowie dem Startpunkt eines Buchstabens immer größer als null und unterliegt während der abgesetzten Bewegung keiner Änderung.

5.5 Experimente

In diesem Abschnitt werden die Experimente zur Buchstabenmodellierung zusammen mit den Ergebnissen beschrieben. Zuerst wird die Erkennung einzelner Buchstaben, dann die Erkennung von Wörtern und abschließend die Erkennung ganzer Sätze beschrieben. Die Auswertung erfolgt auf den Daten des Sensorstiftes und auf den Airwriting-Daten. Die Initialisierung und das Training aller Systeme erfolgt grundsätzlich nach dem in Kapitel 4 beschriebenen Verfahren. Veränderungen oder Erweiterungen werden jeweils benannt.

5.5.1 Einzelbuchstabenerkennung

Die Evaluierung der Erkennungsleistung auf Einzelbuchstaben wird hier nur für den Sensorstift detailliert diskutiert. Für den Airwriting Fall wurde das bereits in einer früheren Arbeit des Autors durchgeführt [Amm09], deren Ergebnisse kurz zusammengefasst werden. Die Evaluierung für den Sensorstift wurde auf den Datensätzen Stift.G und Stift.K, also sowohl für Groß- als auch für Kleinbuchstaben durchgeführt. Bis auf das letzte Experiment wurde immer personenabhängig evaluiert, das heißt die Trainings- wie auch die Testdaten kommen von derselben Person. Die Auswertung erfolgt in einer Kreuzvalidierung, bei der die Testmenge immer genau ein Beispiel jedes Buchstabens enthält und die restlichen Wiederholungen der Buchstaben als Trainingsmenge verwendet werden (*leave-one-out*). Alle vorgestellten Ergebnisse verwenden die Merkmalsextraktion D_{ZNorm} für das Drucksignal. In Abschnitt 5.5.1 wird gezeigt, dass diese Methode der Methode D_{Linear} überlegen ist.

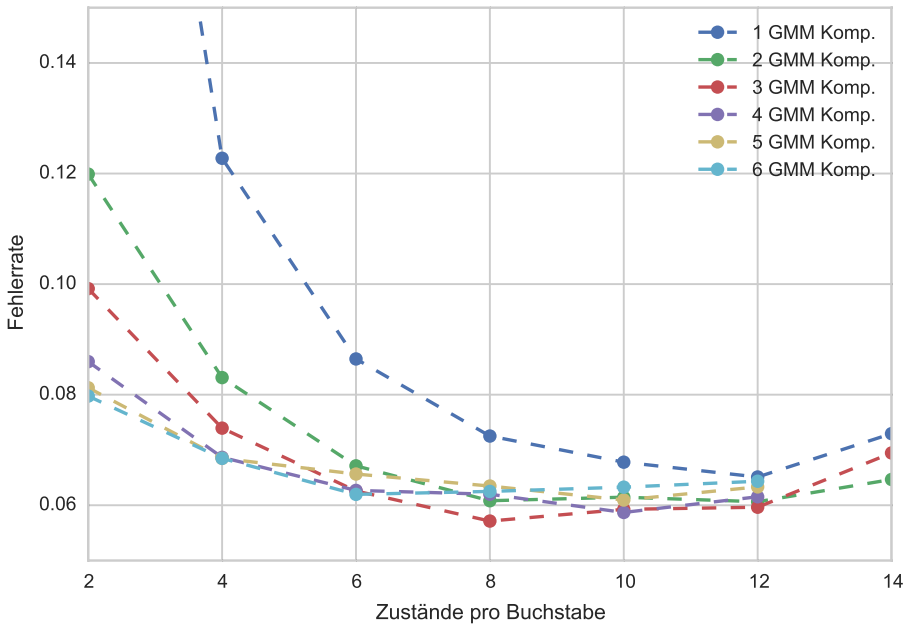


Abbildung 5.3 – Statische Topologien: Ergebnisse der Großbuchstabenerkennung für statische Topologien mit verschiedenen Anzahlen von Zuständen und Komponenten der Gaußschen Mischverteilungen.

Zuerst werden nun auf dem Datensatz der Großbuchstaben Stift.G die unterschiedlichen Methoden zur Wahl der Topologie verglichen und gezeigt, dass eine an die Buchstabenlängen angepasste Topologie eine Verbesserung bringt. Für beide Methoden wird eine Optimierung der freien Modellparameter durchgeführt. Anschließend wird die Erkennung auf den Kleinbuchstaben (Stift.K) und der Kombination von Klein- und Großbuchstaben (Stift.K und Stift.G) evaluiert. Es folgt eine Analyse der typischen Erkennungsfehler sowie der Streuung der Fehlerrate unter den Probanden.

Statische Topologien Im Falle statischer Topologien ist die Anzahl der Zustände für jeden Buchstaben gleich. Um zu evaluieren, welche Erkennungsleistung hiermit erreicht werden kann, wurde die Erkennungsleistung für verschiedene Topologien evaluiert. Zusätzlich wurden auch die Anzahl der Komponenten in Gaußschen Mischverteilungen variiert. Zusammen bestimmen diese zwei Parameter die Anzahl der zu schätzenden Modellparameter. Abbildung 5.3 zeigt die Fehlerraten für die unterschiedlichen Kombinationen von Zustandsanzahl und Komponenten der Mischverteilungen. Für eine ho-

he Anzahl von Zuständen und viele Komponenten sind teilweise zu wenig Daten für eine Schätzung der Parameter vorhanden. Die Initialisierung und das Training kann dann nicht durchgeführt werden, das Programm bricht entsprechend ab. Dies tritt bei mehr als drei Komponenten der Verteilungen und mehr als zwölf Zuständen pro Buchstabe auf. Die Ergebnisse fehlen entsprechend in der Abbildung. Generell gilt, dass bei wenigen Zuständen die Anzahl der Mixturkomponenten einen größeren Einfluss hat als bei einer größeren Anzahl von Zuständen, da die geringe zeitliche Auflösung des Modells mit einer komplexeren Verteilung ausgeglichen werden muss. Im Falle vieler Zustände, also einer hohen zeitlichen Auflösung, ist die Komplexität der Verteilungen geringer und eine kleinere Anzahl von Mixturkomponenten reicht für die Modellierung aus. Die insgesamt niedrigste Fehlerrate wird für acht Zustände und drei Mixturkomponenten erreicht und beträgt 5,7 %. Für den Bereich von acht bis zwölf Zuständen pro Buchstabe und zwei bis fünf Komponenten der Verteilungen werden durchweg niedrige Fehlerraten erreicht.

Für Airwriting wurde im Rahmen der Diplomarbeit [Amm09] und der darauffolgenden Publikation [AGS10] die Erkennung in die Luft geschriebener Einzelbuchstaben für statische Topologien bereits untersucht. Für die personenabhängige Erkennung betrug die Fehlerrate 5,2 %, für die personenunabhängige Erkennung 18,1 %.

Optimierte Topologien Im Falle der optimierten Topologien wird die Anzahl der Zustände für jeden Buchstaben, basierend auf der mittleren zeitlichen Länge der aufgenommenen Buchstaben, bestimmt. Auch für diesen Fall wurde wie für die statischen Topologien, eine Optimierung der freien Modellparameter durchgeführt. Die zwei zu optimierenden Parameter sind die Anzahl der Komponenten in den Gaußschen Mischverteilungen und die mittlere Anzahl von Merkmalsvektoren, die auf einen Zustand entfallen sollen. In Abbildung 5.3 sind die Ergebnisse der Optimierung dargestellt. Hier wird das beste Ergebnis für die Kombination aus zwei Merkmalsvektoren pro Zustand ($\tau = 2$) und fünf Mixturkomponenten erreicht und beträgt 5,1 %. Gegenüber einer statischen Topologie ergibt sich damit eine relative Verbesserung der Fehlerrate um 10,5 %. Die besten Ergebnisse werden für $\tau \in \{2, 3\}$ und ab 2 Komponenten der Mischverteilungen erreicht. Für diese Parameterkombination liegen die Fehlerraten zwischen 5,1 % und 5,5 %.

Analyse In Abbildung 5.5 ist die Konfusionsmatrix für die Erkennung der Großbuchstaben dargestellt. Sie zeigt die aufakkumulierten Verwechslungen

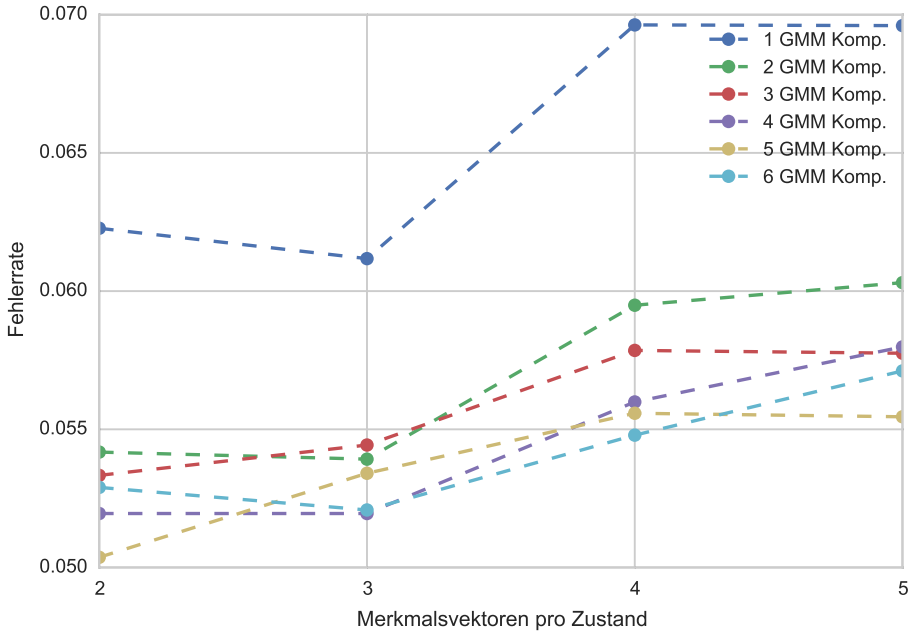


Abbildung 5.4 – Optimierte Topologien: Ergebnisse der Großbuchstabenerkennung für optimierte Topologien mit verschiedenen Anzahlen von Merkmalsvektoren pro Zustand und Komponenten der Gaußschen Mischverteilungen.

für das beste System mit dynamischen Topologien über alle Probanden. Die Zahlen in der Grafik geben den prozentualen Anteil des jeweiligen Paares von Referenz und Hypothese an der Gesamtzahl von Referenzdaten pro Buchstabe an. Die Einträge wurden gerundet, leere Einträge in der Matrix bedeuten entsprechend eine Anzahl von weniger als 0,5 % Verwechslungen. Durch die Rundung addieren sich die Zeilensummen nicht zu den eigentlich korrekten 100 %. Es ist deutlich zu sehen, dass der Hauptanteil der Verwechslungen auf das Paar (P, D) und an zweiter Stelle auf das Paar (A, H) entfällt. Weitere Verwechslungen treten in absteigender Häufigkeit zwischen (G, O), (T, X) sowie (X, Y) auf. Dies deckt sich mit den in [AGS10] analysierten Fehlermustern beim Airwriting. Hier traten häufig für die Buchstabenpaare (P, D), (X, Y), (N, W) sowie (A, H) Fehler auf.

Die Ergebnisse, aufgeschlüsselt nach den einzelnen Probanden, sind in Abbildung 5.6 dargestellt. Die grüne Linie markiert den Mittelwert, die rote Linie markiert den Median. Die Ergebnisse variieren zwischen ca. 1 % und 14 % und haben damit eine vergleichbare Streuung zu den Ergebnissen für das *Airwriting*. In der Publikation des Autors [AGS10] liegen die Ergebnis-

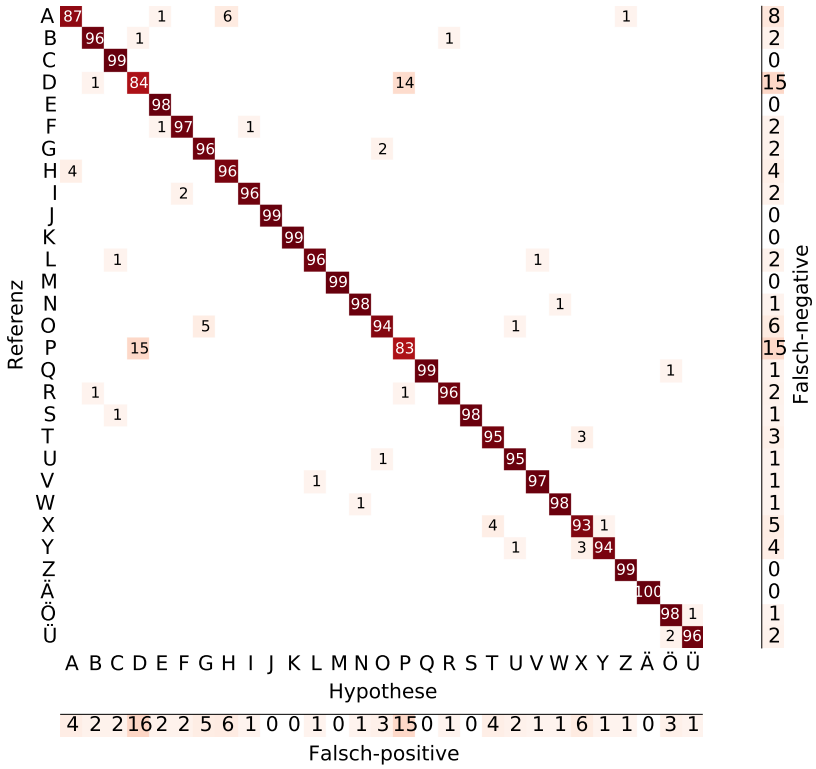


Abbildung 5.5 – Konfusionsmatrizen der Buchstabenerkennung für die Großbuchstaben.

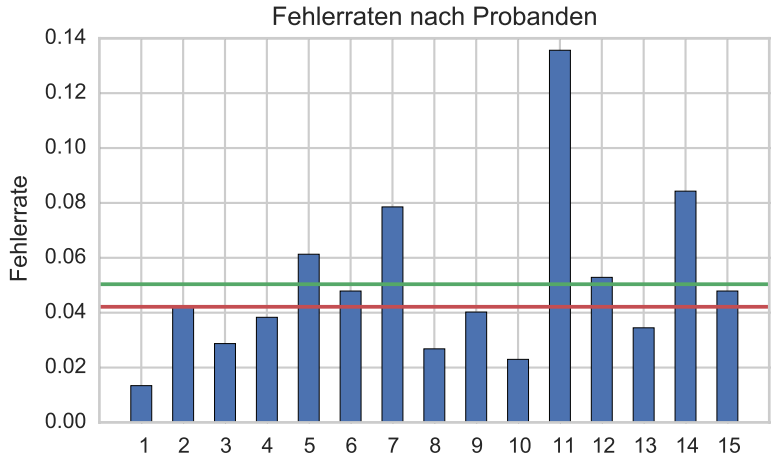


Abbildung 5.6 – Fehlerraten der Großbuchstabenerkennung für die einzelnen Versuchspersonen. Die grüne Linie markiert den Mittelwert, die rote Linie markiert den Median der Ergebnisse.

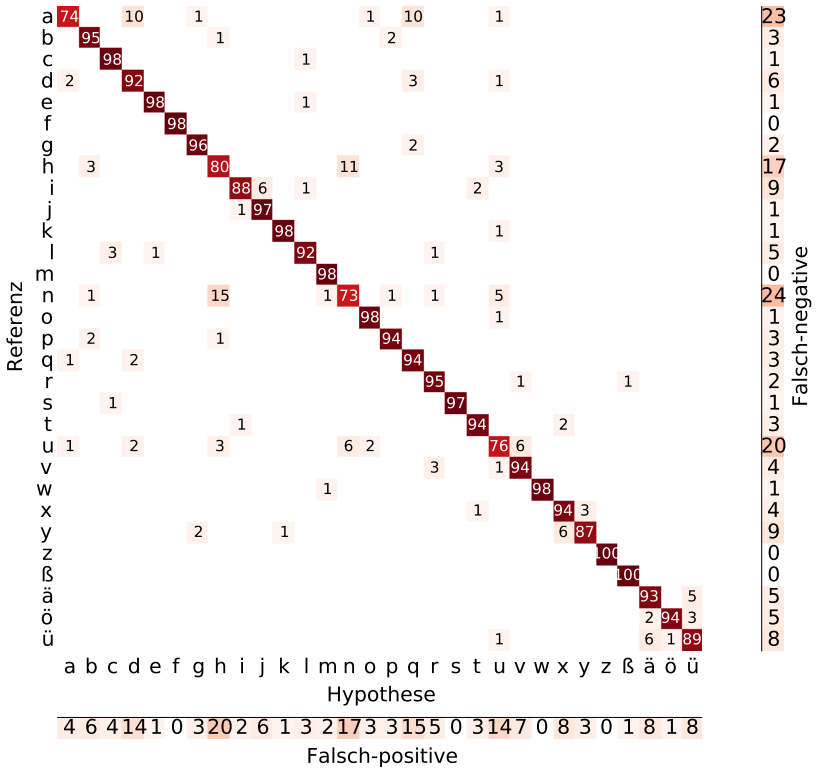


Abbildung 5.7 – Konfusionsmatrizen der Buchstabenerkennung für die Kleinbuchstaben.

se der personenabhängigen Erkennung von Großbuchstaben im Bereich von 1 % bis 11 %. Mögliche Gründe sind einerseits ein Mangel an Konsistenz in der Wiederholung der einzelnen Buchstaben oder eine größere Ähnlichkeit zwischen den Buchstaben durch personenspezifische Schreibvarianten.

Kleinbuchstaben Analog zur Erkennung der Großbuchstaben wurde auch eine Erkennung auf Kleinbuchstaben durchgeführt. Die Anzahl der Klassen beträgt in diesem Fall 30, da im Vergleich zu den Großbuchstaben noch das scharfe S hinzukommt. Die Experimente wurden mit der dynamischen Topologieschätzung sowie einer mittleren Anzahl von zwei Merkmalsvektoren pro Zustand und vier Mixturkomponenten durchgeführt. Damit wird eine Fehlerrate von 8,0 % erreicht. Anhand einer stichprobenhaften Untersuchung wurde sichergestellt, dass die gewählte Parameterkombination auch für Kleinbuchstaben nahe am Optimum liegt. Die Fehlerrate liegt damit um 56 % höher als die für Großbuchstaben. Eine Analyse der in Abbildung 5.7 dargestellten Konfusionsmatrix zeigt eine deutlich größere Streuung der Verwechslungen.

Die meisten Fehlklassifikationen entfallen auf die Buchstaben a, h, n, u, ü, wobei das Paar (n, h) in 13 %, das Paar (a, d) in 6 % der Fälle, das Paar (ä, ü) in 5,5 % und das Paar (n, u) ebenfalls in 5,5 % der Fälle verwechselt wird.

Die Erkennung von Kleinbuchstaben ist aufgrund größerer Ähnlichkeiten zwischen den Buchstaben schwieriger. In der traditionellen automatischen Handschrifterkennung werden ebenfalls niedrigere Fehlerraten für Großbuchstaben als für Kleinbuchstaben erreicht. Dabei liegen die relativen Unterschiede zwischen den Buchstabenklassen in einem vergleichbaren Bereich zu den Resultaten der in dieser Arbeit beschriebenen Experimente (siehe beispielsweise [Koe03]).

Merkmalsextraktion des Drucksignals Sämtliche bisher präsentierten Ergebnisse basieren auf der Methode D_{ZNorm} zur Extraktion der Merkmale aus dem Drucksignal. Mit Methode D_{Linear} wird eine Fehlerrate von 5,8 % auf den Großbuchstaben erreicht, während mit Methode D_{ZNorm}, wie bereits beschrieben, eine Fehlerrate von 5,1 % erreicht wird. Dies entspricht einer relativen Verbesserung um 12 %. Für die Ermittlung der Werte für D_{Linear} wurde wie für D_{ZNorm} eine Optimierung der freien Modellparameter durchgeführt. Die Normalisierung des Drucksignals nach D_{Linear} beinhaltet eine Segmentierung des Signals in Bereiche mit aufgesetztem und abgesetztem Stift anhand zweier Schwellwerte. Obwohl diese Schwellwerte relativ zur Gesamtamplitude definiert sind, können durch die Schwellwertbildung entstehende Fehler einen in Summe negativen Effekt auf die Erkennung haben als die vermeintlich ungenaue Methode D_{ZNorm}. Im Falle von D_{ZNorm} werden die Ungenauigkeiten des Drucksensors teilweise eins-zu-eins an das Erkennungssystem weitergegeben, allerdings können die Fehler nicht verstärkt werden. Der Vergleich der beiden Methoden kann für einen anderen Sensor mit einer besseren Charakteristik anders ausfallen.

5.5.2 Worterkennung

Die Evaluation der Erkennungsleistung auf Wörtern wurde auf dem Datensatz Stift.W durchgeführt. Für jeden Probanden wurden die Modelle auf den Buchstabendaten Stift.G initialisiert und vortrainiert. Danach wurde sowohl direkt auf den Wörtern getestet wie auch eine Kreuzvalidierung auf den Wörtern durchgeführt, wobei immer ein Wort als Testmenge und die verbleibenden Wörter als Trainingsmenge benutzt wurden. Das Vokabular umfasst 912 häufige deutsche Wörter. Ohne ein erneutes Training in der Kreuzvalidierung auf dem Datensatz Stift.W wird eine Fehlerrate von 39 % erreicht. Durch ein

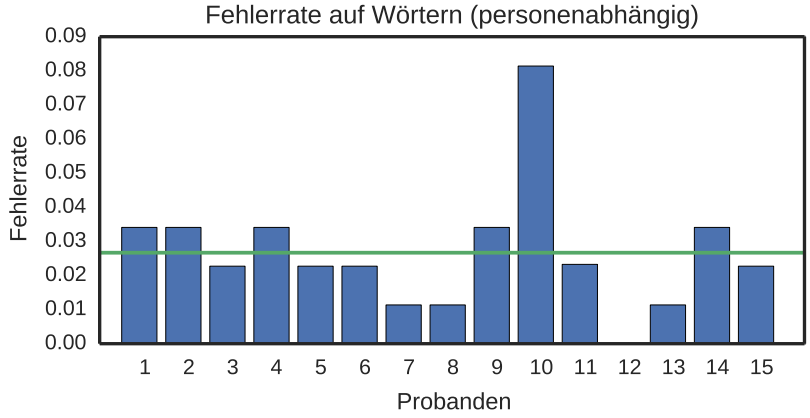


Abbildung 5.8 – Ergebnisse nach Probanden für die personenabhängige Erkennung ganzer Wörter. Die grüne Linie gibt den Mittelwert an.

erneutes Training der Modelle auf den Wortdaten wird eine Fehlerrate von 2,7 % erreicht. Es ist davon auszugehen, dass drei Faktoren wesentlich für die enorme Verbesserung der Fehlerrate sind: 1) Durch ein erneutes Training auf den Wortdaten stehen pro Buchstabe deutlich mehr Trainingsdaten zur Verfügung. 2) Eine Analyse des Schriftbildes der Wörter ergab, dass Buchstaben im Wortkontext teilweise auf andere Art und Weise geschrieben werden als dies für das isolierte Schreiben der Fall ist. 3) Eine Analyse des Schriftbildes der Einzelbuchstaben ergab, dass für isolierte Buchstaben eine sehr hohe Reproduktionsgenauigkeit des Schriftbildes besteht und damit sehr spezifische Modelle entstehen, welche die erhöhte Variabilität der Buchstaben im Kontext der Wörtern nur ungenügend abdecken können. Abbildung 5.8 stellt die Ergebnisse der Evaluierung für die einzelnen Probanden dar. Bis auf einen Ausreißer wurden durchweg Fehlerraten unter 3.5 % erreicht. Damit ist eine personenabhängige Worterkennung mit kleinen Vokabularen bereits praktikabel möglich.

5.5.3 Kontinuierlich geschriebene Sätze

Die Erkennung kontinuierlich geschriebener Sätze wird anhand der *Airwriting* Datensätze Handschuh und Armband evaluiert. Im Unterschied zu den bisherigen Experimenten wird nun noch ein Sprachmodell verwendet, um die Wahrscheinlichkeiten der Wortfolgen zu modellieren. Außerdem wird im Gegensatz zu den bisherigen Experimenten auch der Fall der personenun-

Tabelle 5.1 – Parameter des Sprachmodells.

Abdeckung (1/2/3-gram)	100.00/97.98/64.13
Out-of-vocabulary Rate	0.0
Perplexität	112

abhängigen Erkennung betrachtet. Im Gegensatz zu den Experimenten auf dem Stift-Datensatz enthalten die *Airwriting* Datensätze Wörter und Sätze in englischer Sprache. Entsprechend werden auch keine Modelle für die Umlaute trainiert.

Wörterbuch

Für die Airwriting Experimente werden zwei verschieden große Vokabulare verwendet, um den Einfluss der Größe des Vokabulars zu evaluieren. Die Größe des Vokabulars bestimmt den Suchraum und hat damit Einfluss auf die Fehlerrate des Systems. Das kleine Wörterbuch enthält 986 Wörter, das große Wörterbuch enthält 8231 Wörter. Das kleine Vokabular ist eine Untermenge des großen Vokabulars und beide entstammen einer öffentlich zugänglichen Liste häufiger englischer Wörter[1], welche auf Worthäufigkeiten gesammelter Internetseiten beruhen. Das Vokabular wurde manuell von Abkürzungen bereinigt. Alle Einträge der Länge eins, bis auf „A“ und „I“, die beide häufige Wörter des Englischen sind, wurden gelöscht. Alle Einträge der Länge zwei und drei, welche Abkürzungen (z. B. „OS“) darstellen, wurden gelöscht. Alle Einträge mit einer Länge größer drei blieben unverändert. Letztlich wurden noch alle Wörter der Testsätze dem Vokabular hinzugefügt.

Sprachmodell

Es wird ein 3-Gramm-Sprachmodell verwendet. Das Modell wurde für ein Spracherkennungssystem aus im Internet gesammelter Daten erstellt und beinhaltet 60000 Wörter [Sch02]. Die relevanten Parameter des Sprachmodells sind in Tabelle 5.1 gegeben. Die N-Gramm-Abdeckung beträgt für beide Vokabulare 100 %, das heißt das Sprachmodell beinhaltet alle Wörter des Wörterbuchs. Auch die Bigramm Abdeckung ist mit ungefähr 98% noch sehr hoch. Die Perplexität auf den Testsätzen beträgt 112.

[1] wortschatz.uni-leipzig.de/html/wliste

Initialisierung und Training

Die Initialisierung erfolgt über das in 2.3.2 beschriebene Flatstart Verfahren auf dem Buchstabendatensatz Handschuh.B. Die Modelle werden sukzessive auf den Buchstabendaten (Handschuh.B), den Wortdaten (Handschuh.W) und letztlich innerhalb einer Kreuzvalidierung auf den Satzdaten (Handschuh.S) trainiert.

Personenabhängige Erkennung

Die personenabhängige Fehlerrate gibt die Qualität eines auf den jeweiligen Schreiber angepassten Systems wieder. Zudem kann sie als obere Schranke für die personenunabhängige Erkennung dienen. Gerade im Bereich des *Wearable Computings*, in dem davon ausgegangen werden kann, dass Geräte nur von einem Benutzer genutzt werden, kann der personenabhängige Fall als Standard angesehen werden. Eine typische Strategie zur Erzeugung eines personenabhängigen Systems in einem Produkt wäre, das Gerät mit einem personenunabhängigen Modell auszuliefern und im Laufe des Gebrauchs auf die eigene Handschrift automatisch anzupassen.

Für die Evaluation wurde für jeden Probanden eine 10-fache Kreuzvalidierung auf der Menge der 80 Sätze dieses Probanden aus dem Handschuh.S-Datensatz durchgeführt. Davor wurden die Modelle auf dem Datensatz Handschuh.B initialisiert und auf den Datensätzen Handschuh.B und Handschuh.W trainiert. Es handelt sich also nicht um ein vollständig personenabhängiges System, da die Basismodelle bereits mit Daten anderer Nutzer initialisiert und trainiert wurden. Abbildung 5.9 zeigt die Ergebnisse der personenabhängigen Evaluation zusammen mit den Ergebnissen der personenunabhängigen Evaluation, die im nächsten Abschnitt beschrieben wird. Die mittlere Wortfehlerrate über die Schreiber beträgt 3 %.

Personenunabhängige Erkennung

Die personenunabhängige Fehlerrate wurde ebenfalls mittels Kreuzvalidierung bestimmt. Die Satzdaten jeweils eines Probanden wurden als Testmenge verwendet und die Daten der anderen acht Personen als Trainingsmenge. Wie im personenabhängigen Fall wurden die Modelle vorher auf den Datensätzen Handschuh.B und Handschuh.W initialisiert und trainiert. Die Menge der Probanden im Datensatz Handschuh.S ist disjunkt zur Menge der Probanden in den anderen beiden Datensätzen (Handschuh.B und Handschuh.W). Diese

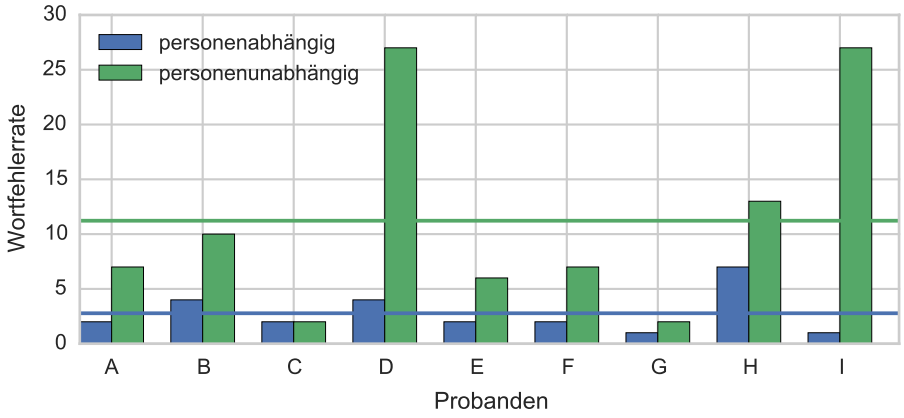


Abbildung 5.9 – Ergebnisse der personenabhängigen und personenunabhängigen Evaluation.

Evaluation entspricht also dem Fall, dass eine Person das System benutzt, ohne dass das System bereits Wissen über den individuellen Schreibstil der Person besitzt.

Im Vergleich mit den personenabhängigen Ergebnissen ist deutlich zu sehen, dass die Probanden mit hoher personenunabhängiger Fehlerrate in sich konsistent geschrieben haben, da sie personenabhängig gute Ergebnisse erzielen. Für den Probanden D ist durch Beobachtung des Experimentes bekannt, dass er das Handgelenk während des Schreibens nicht immer starr gehalten hat. Das Schreiben erfolgte teilweise durch Rotation des Handgelenks. Dies führt einerseits zu unterschiedlichen Beschleunigungs- und Drehratenverläufen im Vergleich zu einer starren Handgelenkshaltung. Anderseits gilt die Annahme der konstanten Orientierung über die Schreibdauer nicht, wodurch die verwendete Normalisierung den Einfluss der Gravitation nur noch ungenügend kompensieren kann. Dies sind mögliche Gründe für die hohe Wortfehlerrate. Für Probanden I gibt es keinen offensichtlichen Grund, warum die Fehlerrate in der personenunabhängigen Erkennung so hoch ist. Ein möglicher Grund ist eine von den anderen Probanden unterschiedliche Schreibweise vieler Buchstaben. Aus der Analyse des Handschuh.B-Datensatzes ist bekannt, dass selbst für Druckschrift viele unterschiedliche Schreibvarianten zwischen unterschiedlichen Schreibern bestehen (siehe Abschnitt 3.3.2). Da die Trajektorie nicht bekannt ist, lässt sich dies aber nicht überprüfen.

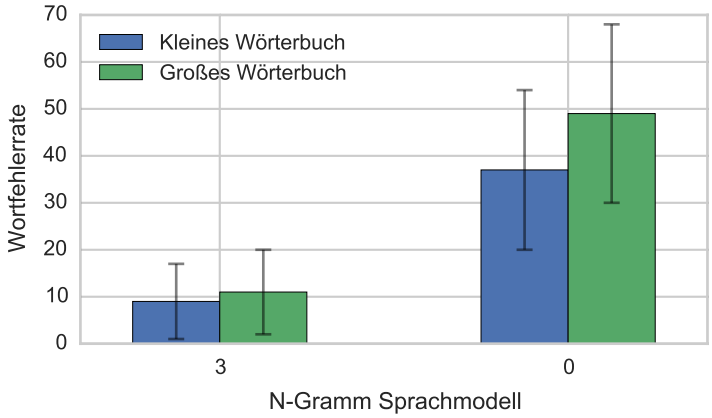


Abbildung 5.10 – Einfluss der Wörterbuchgröße und des Sprachmodells auf die Erkennungsgenauigkeit.

Wörterbuchgröße und Sprachmodell

Sowohl die Größe des Wörterbuches als auch das Sprachmodell haben Einfluss auf die Erkennungsgenauigkeit. Ein größeres Wörterbuch erhöht die Anzahl möglicher Wörter und erhöht damit die Wahrscheinlichkeit von Fehlern. Auf dem großen Wörterbuch wurde, wie bereits beschrieben, eine mittlere Wortfehlerrate von 11 % erreicht. Für das kleine Wörterbuch wird eine Wortfehlerrate von 9 % erreicht. Zwischen kleinem und großen Wörterbuch gibt es demnach eine Steigerung der Fehlerrate um den Faktor 1,2 bei einer Vergrößerung des Wörterbuches um den Faktor 8,3. Die Fehlerrate skaliert also sehr gut mit der Größe des Wörterbuches. Der Einfluss des Sprachmodells auf die Erkennung wurde ebenfalls untersucht. Hierfür wird die Erkennungsleistung mit dem 3-Gramm Sprachmodell mit einem 0-Gramm Sprachmodell verglichen. Bei einem 0-Gramm Modell ist die Wahrscheinlichkeit über alle Wörter im Wörterbuch gleich verteilt und unabhängig von der Historie. In diesem Fall ist also ausschließlich das motorische Modell und das Wörterbuch für die Erkennung relevant. Die Verwendung eines 0-Gramm Sprachmodells erlaubt die Analyse der Erkennungsgenauigkeit, falls keine zusätzlichen Informationen über die Struktur der Sprache vorhanden sind. Für das kleine Wörterbuch wird mit einem 0-Gramm Sprachmodell eine Wortfehlerrate von 37 % , für das große Wörterbuch eine Wortfehlerrate von 49 % erreicht. Abbildung 5.10 zeigt die Ergebnisse der Evaluation für die Verwendung des großen und kleinen Vokabulars und eines 3-gram und 0-gram Sprachmodells.

Die Verwendung eines Sprachmodells erscheint wesentlich für das Erreichen niedriger Fehlerraten bei kontinuierlichen Sätzen.

Erkennung mit Armband

Die bisher beschriebenen Experimente zum Airwriting wurden auf Daten evaluiert, die mit dem Sensorhandschuh aufgenommen wurden. Auf diesen Daten konnte in den vorangegangenen Abschnitten gezeigt werden, dass eine Erkennung von Airwriting mit geringen Fehlerraten möglich ist. Die Erfassung selbst ist aber durch die Verwendung des Handschuhs wenig praktikabel. Aus diesem Grund wird auch eine Bewegungserfassung mit dem in Abschnitt 3.1 beschriebenen Armband untersucht. Um die aufwändige Aufnahme von Einzelbuchstaben für die Initialisierung der Modelle zu vermeiden, wurde eine Transformation der Sensordaten des Armbandes in das Koordinatensystem des Handschuhs durchgeführt (siehe Abschnitt 3.3.1). Durch die folgenden Experimente wird neben der Bewegungserfassung mit dem Armband auch die Tauglichkeit des Ansatzes zur Transformation des Koordinatensystems untersucht.

Das Training erfolgte auf dem Handschuh-Datensatz. Die Modelle wurden dabei analog zu den Experimenten in Abschnitt 5.5.3 auf dem Datensatz Handschuh.B initialisiert und auf Handschuh.B und Handschuh.W trainiert. Anschließend wurden die Modelle weiter auf dem Datensatz Handschuh.W trainiert. Damit sind die finalen Modelle mit dem in Abschnitt 5.5.3 evaluierten System vergleichbar.

Abbildung 5.11 zeigt die Ergebnisse. Die mittlere Wortfehlerrate liegt bei 25 %. Die Verteilung über die acht Probanden variiert allerdings stark. Für fünf Probanden wurden Fehlerraten im Bereich 1 % bis 11 % erreicht, also in einem vergleichbaren Rahmen zu den Experimenten auf dem Handschuh.S-Datensatz (siehe Abschnitt 5.5.3). In diesen Fällen konnte mit dem Armband und der Koordinatensystemtransformation direkt ein gutes Ergebnis erzielt werden. Für drei Probanden traten allerdings Fehlerraten zwischen 52 % und 61 % auf. Unter der Annahme dass nicht der Sensor oder das Armband selbst das Problem sind, kommen mehrere Ursachen für die hohen Fehlerraten in Frage: Wie in Abschnitt 5.5.3 bereits beschrieben, kann der individuelle Schreibstil ein Grund sein, falls sich dieser stark von den Schreibern der Trainingsmenge unterscheidet. Ein weiterer bereits genannter möglicher Grund ist die Ausführung der Bewegung aus dem Handgelenk heraus. Dies hat im Falle einer Sensorpositionierung am Handgelenk einen größeren negativen Einfluss als bei einer Sensorpositionierung an der Hand,

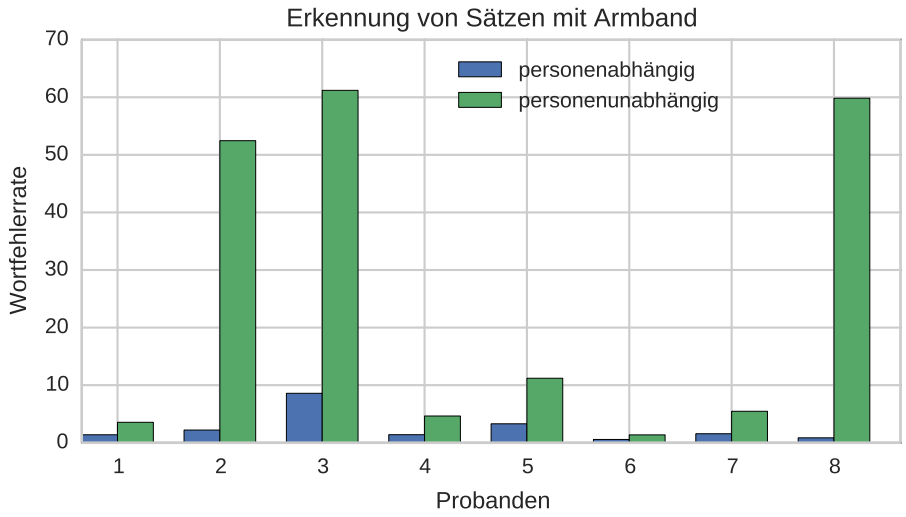


Abbildung 5.11 – Ergebnisse der Satzerkennung auf dem Armband-Datensatz für den personenunabhängigen Fall (blau) und den personenabhängigen Fall (grün).

da bei einer Bewegung der Hand aus dem Handgelenk heraus, der Unterarm häufig eine gegenläufige Bewegung vollführt. Auch die relative Positionierung der Hand zum Unterarm kann ein Grund für erhöhte Fehlerraten sein, da die Transformation des Sensorkoordinatensystems nicht für jede Person individuell, sondern global berechnet wurde. Dies führt bei Unterschieden in der Handhaltung zu einem unterschiedlichen Merkmalsraum für Trainings- und Testdaten. Es ist anzunehmen, dass letztlich eine Kombination verschiedener Gründe zu den hohen Fehlerraten führt. Eine exakte Analyse ist auf Basis der Daten nicht möglich. Hierzu wäre ein Datensatz nötig, der parallele und zeitsynchrone Aufnahmen des Handschuhs und des Armbandes enthält. Dann könnten die mit der Sensorpositionierung zusammenhängenden Gründe von den durch den Schreibstil bedingten Gründen abgegrenzt werden.

Durch eine nachgelagerte personenabhängige Evaluation der Erkennungsleistung, ist es allerdings möglich, zu überprüfen, ob grundsätzliche Probleme bei der Datenerfassung mit dem Sensor oder Armband bestehen und ob die Modelle selbst im Falle der hohen Fehlerraten das Training personenspezifischer Modelle erlauben. Hierfür wurde für jeden Probanden eine 10-fache personenabhängige Kreuzvalidierung durchgeführt. Als Basismodelle wurden die gleichen Modelle wie für die gerade beschriebene personenunabhängige Evaluation verwendet. Diese wurden dann aber im Rahmen der Kreuzvali-

dierung noch einmal auf den Daten der Testperson trainiert. Die Ergebnisse sind durch die grünen Balken in Abbildung 5.11 für die einzelnen Probanden angegeben. Die mittlere Fehlerrate beträgt 2,5 % und sinkt auch für die Probanden, die im personenunabhängigen Fall eine sehr hohe Fehlerrate hatten, auf zu den anderen Probanden vergleichbare Werte. Damit kann ausgeschlossen werden, dass der Sensor oder das Armband die Ursache für die hohen Fehlerraten ist. Außerdem zeigt das Ergebnis, dass die auf dem Handschuh-Datensatz trainierten Modelle auch bei hohen Fehlerraten genutzt werden können, um personenabhängige Systeme für den Armband-Datensatz zu erstellen, welche niedrige Fehlerraten aufweisen. Damit ist für die Erkennung von Airwriting mit dem Armband keine Neuinitialisierung auf Aufnahmen von Einzelbuchstaben nötig. Für die Mehrzahl der Probanden wurde zudem bereits ohne die Erstellung eines angepassten Systems eine gute Fehlerrate erreicht.

5.5.4 Zusammenfassung

Die in diesem Kapitel beschriebenen Untersuchungen zeigen, dass eine Schrifterkennung mittels Inertialsensoren möglich ist. Dies gilt gleichermaßen für Airwriting als auch das Schreiben mit einem herkömmlichen Stift. Beide Ansätze erlauben neuartige Schnittstellen in unterschiedlichen Anwendungskontexten. Dabei werden insbesondere für den personenabhängigen Fall auf Wörtern und Sätzen Fehlerraten erreicht, die zwischen 2 % und 3 % liegen und damit bereits niedrig genug für praktische Anwendungen sind.

Für die anvisierten Anwendungen sind insbesondere die personenabhängigen Fehlerraten relevant, da davon auszugehen ist, dass *Wearable Computing* Geräte im Wesentlichen von einer Person benutzt werden. Dies gilt auch für mögliche Anwendungen des Stiftes. Die Geräte könnten mit einem personenunabhängigen Modell ausgeliefert werden, welches dann entweder unüberwacht oder überwacht, mittels der Aufnahme einer kleinen Trainingsdatenmenge, an den Nutzer angepasst wird.

Für das Airwriting konnte zudem gezeigt werden, dass auch eine personenunabhängige Erkennung kontinuierlich geschriebener Sätze möglich ist. Hier lag die Fehlerrate bei 11 %. Auch die Erkennung mit einem Armband ist mit vergleichbaren Fehlerraten möglich. Hierfür können die auf dem Handschuh-Datensatz trainierten Modelle verwendet werden, es müssen keine Aufnahmen von Einzelbuchstaben für eine Initialisierung vorliegen. In einigen Fällen werden allerdings sehr hohe Fehlerraten beobachtet. Eine personenabhängige

Anpassung auf Basis der personenunabhängigen Modelle senkt die Fehlerrate aber auch in diesem Fall auf 2,5 % ab.

Eine Analyse der Klassifikation von Einzelbuchstaben ergab, dass die Menge der Großbuchstaben mit einer Fehlerrate von 5,1 % deutlich besser diskriminierbar ist als die Menge der Kleinbuchstaben mit einer Fehlerrate von 8,0 %. Insbesondere die Menge der Großbuchstaben ist bis auf einige schwierig zu unterscheidende Paare von Buchstaben gut diskriminierbar. Mit Abstand die meisten Fehlklassifikationen treten dabei für das Paar (D, P) auf, zwei Buchstaben, die sich nur in der Höhe des Bogens unterscheiden. Bei den Kleinbuchstaben ist beispielsweise das Paar (h, n) schwierig zu unterscheiden. Weiter wurde gezeigt, dass die Fehlerrate durch eine an die zeitliche Dauer der Buchstaben angepasste HMM-Topologie um 10,5 % verringert werden kann. Der Drucksensor des Stiftes hat einen relativ hohen Fehler. Es wurden zwei Methoden zur Normalisierung und Merkmalsextraktion verglichen und gezeigt, dass mit einer z-Normalisierung eine um 12 % niedrigere Fehlerrate erreicht wird als mit einer schwellwertbasierten Segmentierung, welche die Schreibbewegung in an- und abgesetzte Phasen unterteilt.

Kapitel 6

Einzelstrichbasierte Erkennung

In diesem Kapitel wird die Modellierung auf Einzelstrichebene vorgestellt. Die Vorteile einer solchen Modellierung werden dargelegt und der Effekt der Kontextabhängigkeit der Bewegungssignale im Falle einer Zerlegung in Einzelstriche erklärt. Neben einer grundsätzlichen Validierung des Ansatzes wird gezeigt, dass eine kontextabhängige Modellierung zu einer deutlichen Reduktion der Fehlerrate im Vergleich zu einer kontextunabhängigen Modellierung führt. Weiter wird gezeigt, dass die Strichmodellierung einen Transfer auf ungesehene Zeichenmengen erlaubt und somit erstmalig die Erkennung von Zeichen möglich macht, die nicht in den Trainingsdaten vorkommen. In einer abschließenden Untersuchung wird der Ansatz für die personenunabhängige Buchstabenerkennung evaluiert. Die Strichmodellierung erreicht eine vergleichbare Fehlerrate wie die Modellierung auf Buchstabenebene, ist allerdings deutlich mächtiger, da sie eine einfache Repräsentation von Schreibvarianten sowie die Erkennung ungesehener Zeichen erlaubt. Martin Rombach hat im Rahmen seiner Bachelorarbeit die Vorarbeiten für die in diesem Kapitel beschriebene Modellierung erarbeitet.

6.1 Striche als atomare Einheiten

Buchstaben können als eine Sequenz von Einzelstrichen dargestellt werden. Einzelstriche stellen daher noch elementarere Einheiten dar als Buchstaben.

Um die Modellierung möglichst einfach und generalisierend zu gestalten, soll in diesem Abschnitt daher die Modellierung durch Einzelstriche untersucht werden. Ein Strich ist dabei als elementare Bewegungseinheit definiert, aus denen sich komplexere Zeichen zusammensetzen lassen. Eine derartige Modellierung hat gegenüber einer buchstabenbasierten Modellierung folgende Vorteile:

- Generalisierbarkeit auf neue Schriftzeichen: Durch eine Modellierung auf Strichebene können theoretisch alle Zeichen erkannt werden, welche sich aus der Menge der gewählten primitiven Striche zusammensetzen lassen. In dieser Arbeit wird dies beispielsweise anhand der Erkennung von Zahlen und Symbolen mit ausschließlich auf Buchstaben trainierten Modellen gezeigt.

- Einfache Modellierung von Varianten: Schreibvarianten von Buchstaben, wie sie in 3.3.2 beschrieben sind, können durch eine Modellierung auf Strichebene einfach abgedeckt werden. Insbesondere ist es nicht nötig, für jede Variante Trainingsdaten zur Verfügung zu haben. Dieser Ansatz wird in der vorliegenden Arbeit mit der Modellierung auf Buchstabenebene verglichen und es wird gezeigt, dass damit eine vergleichbare Erkennungsleistung wie für Einzelbuchstabenmodelle erreicht werden kann.

- Bessere Ausnutzung der Trainingsdaten: Durch eine Modellierung elementarer Einheiten stehen mehr Trainingsdaten für die einzelnen Einheiten zur Verfügung.

Allerdings hat auch eine Modellierung auf Buchstaben- oder Zeichenebene Vorteile gegenüber einer Modellierung auf Strichebene. Ein ideales Buchstabenmodell ist auf die Abbildung genau dieses Buchstabens spezialisiert. Strichmodelle dagegen müssen einzelne Striche modellieren, die in unterschiedlichen Buchstaben verwendet werden. Ein Beispiel dafür wären die vertikalen Linien, wie sie in den Buchstaben F, E, T, usw. vorkommen. Daher ist davon auszugehen, dass Strichmodelle stärker generalisieren müssen und entsprechend weniger spezifisch für einzelne Zeichen sind.

Die zwei Ansätze, Strichmodellierung und Buchstabenmodellierung, lassen sich verbinden. Eine Modellierung auf Strichebene kann in eine Modellierung auf Zeichenebene überführt werden, um durch ein erneutes Training eine spezifischere Modellierung auf Zeichenebene zu erhalten. In umgekehrter Richtung ist dies allerdings nicht möglich, die Strichmodellierung ist daher flexibler.

In der Spracherkennung hat sich die Modellierung elementarer Phoneme gegenüber einer Modellierung komplexerer Einheiten, wie beispielsweise Wörtern, überlegen erwiesen. Auch in der traditionellen Handschrifterkennung wird dieser Ansatz verfolgt [HLB00, PS00, CK04, AMG07]. Da hier meist die Erkennung von kursiv geschriebener Schrift im Fokus steht, ist eine einfache Definition eines elementaren Striches nicht möglich. In [HLB00] werden die elementaren Modellierungseinheiten nicht manuell definiert, sondern datengetrieben bestimmt. Dieser Ansatz erfordert allerdings eine große Menge an Trainingsdaten und lässt keine Erkennung ungesehener Zeichen zu. Einen ähnlichen Ansatz wie in dieser Arbeit verfolgen Artieres et al. in [AMG07]. Zeichen werden aus einer Menge von 36 geraden und gebogenen Strichen zusammengesetzt um beliebige Buchstaben und Symbole zu erkennen. In allen Fällen steht als Eingabe allerdings die exakte Trajektorie zur Verfügung und mithilfe der Strichmodelle werden Segmente dieser Eingangstrajektorie modelliert. Im Falle der Schrifterkennung mit Inertialsensoren müssen die Segmente dagegen anhand ihrer charakteristischen Beschleunigung und Winkelgeschwindigkeit modelliert werden. Dass auch dies möglich ist, wird erstmalig mit den im Rahmen dieser Arbeit vorgestellten Experimenten gezeigt.

6.1.1 Menge der primitiven Striche

Die Buchstaben und Zahlen des lateinischen Alphabets bestehen im Wesentlichen aus einer Aneinanderreihung von Strichen unterschiedlicher Form. Ein N besteht demnach aus einem vertikalen Strich, gefolgt von einem schrägen Strich und zuletzt wieder einem vertikalen Strich. Für eine Modellierung aller Zeichen des Alphabets auf Strichebene ist demnach eine Menge primitiver Striche notwendig, aus denen sich alle Zeichen der Zieldomäne zusammensetzen lassen. Als Zieldomäne wurden das lateinische Alphabet, die Ziffern 0-9 sowie eine Reihe einfacher Bewegungen und verschiedener Symbole (siehe auch Abbildung 3.3, Seite 35) gewählt. Eine abgeschlossene Menge von primitiven Strichen für Schriftzeichen kann nur bedingt angegeben werden. Sobald ein Schreiber von der exakten Reproduktion der Druckschrift abweicht und beispielsweise Elemente der kursiven Schreibschrift miteinfließen lässt, Übergänge abrundet oder die Neigung der Schrift ändert, ändert sich auch Form und Richtung der einzelnen Striche. Dementsprechend kann eine Auswahl von primitiven Strichen immer nur eine möglichst gute Annäherung an die vollständige Abdeckung, der in den Daten vorkommenden Bewegungen, sein.

Zeichen	Strichsequenz
A	$G_{or}\ G_{ur}\ G_{ol}\ G_r$
A(1)	$G_{ul}\ G_{or}\ G_{ur}\ G_{ol}\ G_r$
E	$G_u\ G_r\ G_{ol}\ G_r\ G_{ol}\ G_r$
P	$G_u\ G_o\ K_{ru}$
D	$G_u\ G_o\ K_{ru}$
1	$G_{ol}\ G_u$

Tabelle 6.1 – Beispielhafte Strichsequenzen für einzelne Buchstaben.

Als Menge von primitiven Strichen werden acht gerade Striche und acht Kreissegmente festgelegt. Diese 16 Striche werden im weiteren Verlauf auch als Primitive bezeichnet. Sie werden in Kapitel 3.2.3 erläutert und in Abbildung 3.4 mit ihren Bezeichnern eingeführt. Mit dieser Auswahl können die gewählten Zieldomänen weitgehend abgedeckt werden. Eine Auswahl stellt naturgemäß immer einen Kompromiss zwischen der Anzahl von Primitiven und der Abdeckung der Zieldomäne dar. Tabelle 6.1 gibt beispielhaft für einige Zeichen die entsprechende Sequenz der Striche an. Varianten sind mit einer in Klammern angegeben Nummer gelistet. Die ohne Klammern angegebene Variante gibt dabei die Grundform an. Für die Erkennung werden Grundform und Varianten gleichwertig behandelt.

Aufgrund der in Abschnitt 2.2 beschriebenen Eigenschaften von Inertialsensoren ist es, im Gegensatz zur Bewegungsrichtung, schwierig, die Länge der Bewegung zu bestimmen. Daher werden keine Striche mit unterschiedlichen Längen oder Radien definiert. Dadurch werden allerdings beispielsweise die Buchstaben P und D mit der gleichen Strichsequenz modelliert. Eine Unterscheidung zwischen P und D ist dann nicht mehr möglich, beide Buchstaben bilden eine gemeinsame Klasse, das heißt die Unterscheidung muss anhand linguistischem Wissens auf Wortebene erfolgen. Aus den Experimenten der Modellierung auf Buchstabenebene (siehe Kapitel 5) ist bereits bekannt, dass diese Unterscheidung auch mit buchstabenspezifischen Modellen schwierig ist, die Ambiguität allerdings gut auf Wortebene aufgelöst werden kann. Eine Erweiterung der Strichmenge um größenspezifische Modelle erscheint wenig aussichtsreich, würde aber die Menge der atomaren Modelle erhöhen. Grundsätzlich wäre eine systematische Untersuchung unterschiedlicher Mengen von Primitiven interessant und könnte im Rahmen zukünftiger Arbeiten durchgeführt werden. Eine vollständige Auflistung aller Zeichen und Varianten mit ihren verwendeten Strichsequenzen ist in Anhang A.4 gegeben.

6.1.2 Kontextabhängigkeit der Einzelstriche

Durch die Modellierung der sehr kurzen primitiven Striche gewinnt ihre Kontextabhängigkeit an Bedeutung. Der Kontext eines Strichs ist durch den vorangegangenen und den nachfolgenden Strich definiert. Der Kontext eines Primitivs P mit Vorgänger P_V und Nachfolger P_N wird durch die Notation $P(P_V|P_N)$ angegeben. Die Abhängigkeit vom Kontext besteht in zwei wesentlichen unterschiedlichen Effekten. Erstens werden Übergänge zwischen Strichen je nach Kontext unterschiedlich ausgeführt. Beispielsweise werden Ecken, die durch Verkettung zweier gerader Striche entstehen, möglicherweise abgerundet. Zweitens hängt das Beschleunigungssignal in hohem Maße vom Kontext ab, da sich bei einer Richtungsumkehr der Bewegung das Vorzeichen der Beschleunigung nicht ändert. Die Abbremsbeschleunigung des ersten Teils der Bewegung geht direkt in die Vorwärtsbeschleunigung des zweiten Teils der Bewegung über. Dieser Effekt ist beispielsweise bei der direkten Abfolge eines Striches nach oben, gefolgt von einem Strich nach unten, zu beobachten. Findet eine Richtungsänderung orthogonal zur ersten Bewegung oder eine Pause nach der ersten Bewegung statt, fällt die Beschleunigung auf null zurück. Dies führt dazu, dass sich das Beschleunigungssignal eines isoliert ausgeführter Striches, also aus der Ruheposition startend und in der Ruheposition endend, stark vom gleichen Strich unterscheiden kann, wenn dieser in eine Strichsequenz eingebettet ist. Der Effekt ist in Abbildung 6.1 für das Primitiv G_o, also einem Strich nach oben, dargestellt. Für dieses Primitiv werden die Signale für die zwei Vorkommnisse im Buchstaben N, also im Kontext $G_o(Ruhe|G_{ur})$ und $G_o(G_{ur}|Ruhe)$, sowie als isolierte Ausführung, also im Kontext $G_o(Ruhe|Ruhe)$, dargestellt. Die Unterschiede im Beschleunigungssignal sind deutlich zu sehen. Motiviert durch dieses Verhalten wird zusätzlich zu einer kontextunabhängigen Strichmodellierung auch eine kontextabhängige Modellierung der Signale untersucht.

6.1.3 Initialisierung der Einzelstrichmodelle

Analog zu der in Kapitel 2.3.2 beschriebenen Initialisierung der Modelle auf den Buchstaben müssen die atomaren Striche auf Daten von einzelnen Strichen initialisiert werden. Aufgrund der im vorherigen Abschnitt beschriebenen Kontextabhängigkeit der Einzelstriche ist es wichtig, dies nicht ausschließlich auf isoliert aufgenommenen Beispielstrichen durchzuführen. Die Initialisierung der Modelle findet aus diesem Grund auf dem Strich-Datensatz statt. Wie im Absatz 3.2.3 beschrieben, wurden für diesen Datensatz die Trajektorien miterfasst und damit eine Annotierung der Einzelstriche vorge-

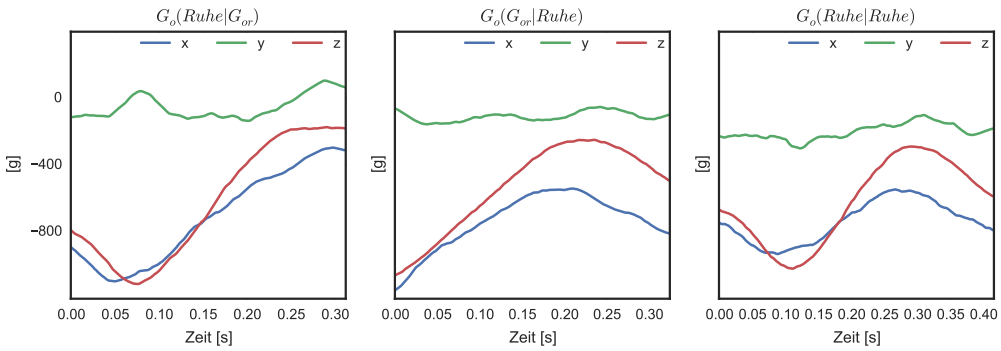


Abbildung 6.1 – Beschleunigungssignale für einen Strich nach oben (G_o) in unterschiedlichen Kontexten. Die zwei linken Grafiken zeigen die Signale der Bewegung, wenn danach oder davor eine Bewegung aus entgegengesetzten Richtungen erfolgte. Sie sind einer Aufnahme des Buchstabens N entnommen und zeigen den ersten und letzten Strich des Buchstabens. Die rechte Grafik zeigt die Signale eines Strichs nach oben, wenn vorher und nachher keine Bewegung erfolgte. Je nach Kontext führt die gleiche Bewegung zu unterschiedlichen Signalen.

nommen. Damit stehen die exakten Anfangs- und Enddaten der Einzelstriche sowie die Information, in welchem Kontext die Striche vorkommen, zur Verfügung.

6.2 Vorverarbeitung

Die Aufnahmen der Primitive sind sehr kurz und weisen häufig nur Bewegung in einer Dimension auf, beispielsweise für die einzeln ausgeführten Primitive. Eine auf allen Kanälen unabhängig ausgeführte Varianznormalisierung führt zu einer Verzerrung der Signale in den Dimensionen, welche wenig Bewegungsaktivität enthalten. Im Extremfall wird das Sensorrauschen auf den gleichen Wertebereich skaliert wie die Signale relevanter Bewegungen. Aus diesem Grund ist diese Form der Normalisierung für die Initialisierung der Einzelstrichmodelle nicht geeignet. Die z-Normalisierung der in Abschnitt 4.2 beschriebenen Vorverarbeitung wird deshalb leicht abgewandelt. Die Varianz der Beschleunigung wird nun über alle Beschleunigungswerte und die Varianz der Drehratenwerte über alle Drehratenwerte normalisiert. Die Mittelwertbereinigung wird weiterhin für alle Dimensionen unabhängig vorgenommen.

6.3 Modellierung von Primitiven

Die Modellierung erfolgt sowohl kontextabhängig, als auch kontextunabhängig. Beide Varianten werden im folgenden beschrieben. Die kontextunabhängige Modellierung entspricht dabei weitgehend dem bereits in Kapitel 5 vorgestellten Schema der Buchstabenmodellierung. Die kontextabhängige Modellierung erfordert eine Erweiterung dieses Schemas.

6.3.1 Kontextunabhängige Modellierung

In der kontextunabhängigen Modellierung gibt es für jedes Primitiv genau ein Modell. Es werden also insgesamt 16 Modelle für die in Kapitel 3.2.3 eingeführten Striche verwendet. Die Modelle werden gemäß dem in Kapitel 4 beschriebenen Verfahren initialisiert und trainiert. Hierfür werden die im Datensatz Strich vorgenommenen Annotierungen benutzt, die für die aufgenommenen Bewegungen die Grenzen der Einzelstriche angeben. Das Wörterbuch gibt dann für jedes Zeichen die Folge von Strichen an, aus denen dieses aufgebaut ist. Für diesen Fall wird entsprechend die durch die Kontextabhängigkeit vorhandene Varianz in den Signalen der einzelnen Primitive durch genau ein Modell abgedeckt.

6.3.2 Kontextabhängige Modellierung

Für die kontextabhängige Modellierung werden Modelle erstellt, die abhängig sind vom vorangegangenen und nachfolgenden Strich. Das zugrunde liegende Verfahren wurde für die automatische Spracherkennung entwickelt [Lee88] und ist beispielsweise in [You08] ausführlich beschrieben. In der Spracherkennung werden Phoneme im Kontext ihrer Vorgänger und Nachfolger modelliert. Wird eine Kontextbreite von eins benutzt, spricht man von Triphonen, es sind aber auch größere Kontextbreiten üblich. In der vorliegenden Arbeit wird für die kontextabhängige Modellierung der Striche eine Kontextbreite von eins verwendet, da davon auszugehen ist, dass die Bewegung eines Striches nur von seinem direkten Vorgänger und Nachfolger abhängt. Eine experimentelle Überprüfung dieser Annahme findet im Rahmen dieser Arbeit allerdings nicht statt. Ein kontextabhängiges Primitiv mit Kontextbreite eins wird, in Anlehnung an den Begriff des Triphons, als Triprimitiv bezeichnet.

Für die verwendete Menge von 16 Primitiven sowie der Ruheposition ergeben sich entsprechend insgesamt theoretisch $17^3 = 4913$ unterschiedliche

Triprimitive. Dieser theoretische Wert ist allerdings zu hoch, da in ihr auch Kombinationen wie $G_o(G_o|G_o)$ enthalten sind, welche einfach durch G_o ersetzt werden können, da die Bewegungen unabhängig von ihrer Länge modelliert werden (siehe Abschnitt 6.1.1). Für alle Primitive, die einen geraden Strich repräsentieren, gilt, dass ihr mehrmaliges Vorkommen in direkter Abfolge durch ein einzelnes Vorkommen ersetzt werden kann. Damit sind die Triprimitive $G_x(G_x|*)$ und $G_x(*|G_x)$ mit $x \in \{o, u, r, l, or, ol, ur, ul\}$ obsolet. Zudem wird das Primitiv *Ruhe* als kontextunabhängig angenommen, damit kommen die Triprimitive $Ruhe(*|*)$ nicht vor. Insgesamt reduziert sich die Anzahl der unterschiedlichen Triprimitive auf 4409. Um ein vollständiges System mit allen Triprimitiven zu erstellen, ist eine genügend große Menge an Trainingsdaten nötig. Diese muss einerseits alle Triprimitive und für jedes Triprimitiv genügend Trainingsdaten zum Schätzen der Modellparameter enthalten. Insbesondere die erste Bedingung ist in der Regel nicht erfüllt, da die vorkommenden Triprimitive von der Zieldomäne, also der gewählten Zeichenmenge, abhängen. Im lateinischen Alphabet und für die alphanumerischen Zeichen kommen beispielsweise keine drei aufeinanderfolgende Bewegungen nach oben oder unten (z. B. $G_u(G_{ul}|G_{ur})$) vor. Der Strich-Datensatz enthält insgesamt nur 121 der möglichen 4409 Triprimitive.

Um eine bessere Ausnutzung der Trainingsdaten und eine bessere Generalisierbarkeit auf unbekannten Triprimitiven zu erreichen, werden ähnliche Modelle zusammengefasst. Ein verbreitetes Standardverfahren hierfür, welches auch in dieser Arbeit zum Einsatz kommt, wird im folgenden Abschnitt beschrieben.

Kontext-Clustering

Das Zusammenfassen ähnlicher Triprimitive wird als Kontext-Clustering bezeichnet. Statt ganze Triprimitive, also die kompletten HMMs zusammenzufassen, wird das Clustering in der Regel auf Ebene der Zustände ausgeführt. Damit können beispielsweise die Anfangszustände aller Triprimitive $G_u(G_o|*)$ eines Striches nach unten mit vorangegangenem Strich nach oben zusammengefasst werden. Die Endzustände dieser Triprimitive richten sich dagegen nach dem Nachfolger und können entsprechend diesem zusammengefasst werden. Die mittleren Zustände könnten über alle Triprimitive $G_u(*|*)$ sehr ähnliche Emissionswahrscheinlichkeiten haben und entsprechend zusammengefasst werden. Damit ergibt sich eine optimale Ausnutzung der Trainingsdaten. Der zweite Vorteil dieses Ansatzes ist die Möglichkeit, ein geeignetes

Modell auch dann zu finden, wenn das Triprimitiv im Test, allerdings nicht in den Trainingsdaten vorkommt.

Der eigentliche Clustering Prozess wird mit einem Top-Down Ansatz durchgeführt. Zu Beginn sind alle Triprimitive unabhängig von ihrem Kontext zusammengefasst und werden dann inkrementell in kleinere Cluster geteilt. Dabei entsteht ein Binärbaum, in dem jedes Blatt einen Cluster von Triprimitiven repräsentiert. Es wird immer der Knoten als nächstes geteilt, dessen Teilung den maximalen Entropieverlust bewirkt. Dabei wird die Menge der möglichen Teilungen in der Regel durch eine vordefinierte Menge möglicher Kontextfragen begrenzt. Die Fragen werden anhand von Vorwissen über die Abhängigkeiten der Primitive untereinander festgelegt. Hierzu werden die Primitive vordefinierten Gruppen zugeordnet, die relevante Eigenschaften der Primitive zusammenfassen. Eine Kontextfrage besteht dann in der Frage, ob das Vorgänger- oder Nachfolgerprimitiv einer bestimmten Gruppe angehört.

Die Gruppen von Primitiven sind in Tabelle 6.2 aufgeführt. Die Wahl erfolgt gemäß dem Wissen über die Kontextabhängigkeit der Striche. Entscheidend ist, wie bereits beschrieben, die Richtung der Bewegung. Im Gegensatz zu den aus geraden Strichen bestehenden Primitiven, beinhaltet ein Kreissegment Bewegung in mehrere Richtungen. Im Falle des Vorgängers ist allerdings nur die Richtung am Ende der Bewegung und im Falle des Nachfolgers nur die Richtung am Anfang der Bewegung entscheidend. Daher werden die Gruppen nach dem Kriterium der vorherrschenden Bewegungsrichtung des Primitives am Anfang oder Ende der Bewegung gebildet. Die Benennung der Gruppen erfolgt nach diesem Schema, die Gruppe “EndeHoch” beinhaltet beispielsweise diejenigen Primitive, die an ihrem Ende eine Bewegung nach oben enthalten.

Bei den Kreissegmenten ist der Anteil der Richtungen am Anfang und Ende der Bewegung nicht klar definierbar, insbesondere da die tatsächlichen Bewegungen nicht perfekten Kreissegmenten entsprechen. Beispielsweise ist für die beim Buchstaben B auftretenden Kreissegmente K_{ru}nicht klar, ob zuerst die Bewegung nach rechts dominiert oder ob von Anfang an bereits eine Bewegung nach unten erfolgt. Um hier eine harte Entscheidung zu vermeiden, werden für beide Varianten Gruppen erstellt (in Tabelle 6.2 mit 1 und 2 markiert). Das Verfahren wählt dann automatisch die besser passende Gruppe über die Maximierung des Entropieverlusts aus.

Um ein kontextabhängiges System zu erstellen wird nach folgendem Verfahren vorgegangen:

Gruppe	Primitive
EndeHoch1	$\{G_o, G_{or}, G_{ol}, K_{ur}, K_{ul}\}$
EndeHoch2	$\{G_o, G_{or}, G_{ol}, K_{ur}, K_{ul}, K_{lo}, K_{ro}\}$
AnfangHoch1	$\{G_o, G_{or}, G_{ol}, K_{or}, K_{ol}\}$
AnfangHoch2	$\{G_o, G_{or}, G_{ol}, K_{or}, K_{ol}, K_{lo}, K_{ro}\}$
EndeUnten1	$\{G_u, G_{ur}, G_{ul}, K_{or}, K_{ol}\}$
EndeUnten2	$\{G_u, G_{ur}, G_{ul}, K_{or}, K_{ol}, K_{lu}, K_{ru}\}$
AnfangUnten1	$\{G_u, G_{ur}, G_{ul}, K_{ur}, K_{ul}\}$
AnfangUnten2	$\{G_u, G_{ur}, G_{ul}, K_{ur}, K_{ul}, K_{lu}, K_{ru}\}$
EndeRechts1	$\{G_r, G_{or}, G_{ur}, K_{lo}, K_{lu}\}$
EndeRechts2	$\{G_r, G_{or}, G_{ur}, K_{lo}, K_{lu}, K_{ur}, K_{or}\}$
AnfangRechts1	$\{G_r, G_{or}, G_{ur}, K_{ro}, K_{ru}\}$
AnfangRechts2	$\{G_r, G_{or}, G_{ur}, K_{ro}, K_{ru}, K_{ur}, K_{or}\}$
EndeLinks1	$\{G_l, G_{ol}, G_{ul}, K_{ro}, K_{ru}\}$
EndeLinks2	$\{G_l, G_{ol}, G_{ul}, K_{ro}, K_{ru}, K_{ul}, K_{ol}\}$
AnfangLinks1	$\{G_l, G_{ol}, G_{ul}, K_{lo}, K_{lu}\}$
AnfangLinks2	$\{G_l, G_{ol}, G_{ul}, K_{lo}, K_{lu}, K_{ul}, K_{ol}\}$
Ruhe	$\{Ruhe\}$

Tabelle 6.2 – Gruppierung der Primitive in Gruppen ähnlicher Bewegungsrichtung am Anfang oder Ende einer Bewegung. Für jede Richtung werden zwei Gruppen gebildet (markiert mit 1 und 2), da die kreisförmigen Primitive je nach Ausführung am Anfang und am Ende sowohl Bewegung in eine als auch zwei Richtungen enthalten können.

1. Training eines kontextunabhängigen Systems (wie in Abschnitte 6.3.1 beschrieben). Das Ergebnis dieses Schrittes ist entsprechend ein HMM-Modell pro Primitiv. Das heißt für jeden Zustand in diesem HMM wird eine Gaußsche Mischverteilung trainiert. Dieses System wird zur Initialisierung des kontextabhängigen Systems in Schritt 2 verwendet.

2. Erstellen eines initialen, kontextabhängigen Systems mit Modellen für alle in den Trainingsdaten vorhandenen Triprimitive. In diesem Schritt werden für alle Triprimitive eigene Modelle trainiert. Dabei werden allerdings ausschließlich die Gewichtsvektoren der Gaußschen Mischverteilungen kontextabhängig modelliert und trainiert. Die Parameter der einzelnen Normalverteilungen selbst sind unabhängig vom Kontext. Alle Triprimitive, die den Kontext eines gemeinsamen Primitives modellieren haben also die exakt gleichen Normalverteilungen in ihren Gaußschen Mischverteilungen. Über die unterschiedlichen Gewichtsvek-

toren ist die Mischverteilung allerdings für jedes Triprimitiv individuell angepasst. Das Resultat dieses Schrittes sind entsprechend kontextabhängige Mixturgewichte bei kontextunabhängigen Verteilungen. Dies erlaubt die Berechnung der Entropie über die Mixturgewichte auf ein und demselben Grundprimitiv (Schritt 3.)

3. Kontext-Clustering der Modelle gemäß des in Abschnitt 6.3.2 beschriebenen Verfahrens. Über einen Top-down Ansatz werden die Zustände der Triprimitive solange in kleinere Gruppen (Cluster) geteilt bis ein Abbruchkriterium erfüllt ist. Dies kann beispielsweise eine maximale Anzahl von Modellen oder eine minimale Anzahl Trainingsdaten pro Cluster sein. Das Ergebnis dieses Schrittes ist eine Einteilung der Zustände der Triprimitive in Cluster abhängig von ihrem Kontext. Jeder dieser Cluster wird nun im finalen Schritt individuell modelliert und trainiert.

4. Training des finalen Systems mit den in Clustern zusammengefassten Zuständen der Triprimitive. Nun werden für jeden Cluster sowohl die Verteilungen als auch die Gewichtsvektoren der Gaußschen Mischverteilungen individuell trainiert. Für jeden Cluster wird also ein eigenständiges Modell trainiert. Das Ergebnis ist ein kontextabhängiges System.

Die Anzahl der resultierenden Modelle wird üblicherweise über ein Abbruchkriterium begrenzt. Das kann beispielsweise die Gesamtanzahl von Modellen oder eine Minimalanzahl von Trainingsdaten für jeden Cluster sein. In dieser Arbeit wird die Gesamtanzahl der Modelle als Abbruchkriterium verwendet, da sich herausgestellt hat, dass in den beschriebenen Experimente die Menge der Trainingsdaten kein limitierender Faktor ist.

6.4 Experimente

Die Evaluierung der Strichmodellierung erfolgte anhand von mehreren Experimenten. In einem ersten Experiment wird gezeigt, dass der Ansatz der Strichmodellierung grundsätzlich funktioniert, sowohl für den kontextunabhängigen als auch für den kontextabhängigen Fall. Die beiden Ansätze werden zudem verglichen und der Einfluss der Anzahl der Modelle für den kontextabhängigen Fall untersucht. Im zweiten Experiment wird untersucht, ob die Strichmodellierung einen Transfer auf eine neue Zieldomäne erlaubt, also dem Test auf Zeichen, die in der Trainingsmenge nicht vorkommen. Auch hier wird der kontextabhängige mit dem kontextunabhängigen Fall vergli-

chen. Im dritten Experiment wird die Modellierung auf Strichebene mit der Modellierung auf Buchstabenebene für die personenunabhängige Erkennung verglichen.

6.4.1 Kontextabhängig vs. Kontextunabhängig

In diesem Experiment wird die grundsätzliche Validität des Ansatzes anhand der Fehlerrate überprüft und zudem die kontextunabhängige mit der kontextabhängigen Modellierung verglichen. Hierfür wird auf dem Strich-Datensatz eine 15-fache Kreuzvalidierung (*leave-one-out*) durchgeführt. Es wird nicht der volle Strich-Datensatz verwendet, da einige identische Zeichen mit unterschiedlicher Semantik vorkommen: die Zahl 0 und der Buchstabe O, das Primitiv G_u und der Buchstabe I und das Primitiv K_{ur} und der Buchstabe U. Es werden jeweils die Daten der Buchstaben verworfen um einen balancierten Datensatz zu erhalten. Damit bleiben 65 unterschiedliche Klassen im Datensatz.

Die Paare (0,6) und (D,P) werden durch die gleichen Strichsequenzen modelliert und sind daher nicht zu unterscheiden. Auch bei perfekter Erkennung der anderen Zeichen ist der Erwartungswert der Fehlerrate damit $2/65 = 0,031$, also 3,1 %. Der tatsächlich von diesen zwei Zeichenpaaren bedingte Fehler schwankt aber zwischen 0 % und 6,2 %, da nicht vorhersehbar ist, welcher der zwei Klassen mit identischer Strichsequenz ein Testbeispiel zugeordnet wird. Daher eignet sich die Erkennungsrate nicht für den Vergleich zwischen den Fehlerraten unterschiedlicher Parameterkombinationen. Aus diesem Grund wird jeweils noch die bereinigte Fehlerrate angegeben, für deren Berechnung die durch Verwechslung innerhalb der Paare (0,6) und (D,P) verursachten Fehler nicht berücksichtigt werden. Der Wert kann daher ausschließlich zum Vergleich unterschiedlicher Experimente verwendet werden und ist keine Abschätzung der realen Fehlerrate.

Um den Einfluss der Anzahl der Modelle zu untersuchen wurde die maximale Anzahl zusätzlicher Modelle als Abbruchkriterium für das Top-Down Clustering variiert. Die Ergebnisse sind in Grafik 6.2 zusammengefasst. Die Zahl der zusätzlichen Modelle gibt dabei an, wie viele Verzweigungen zu dem Entscheidungsbaum hinzugefügt wurden. Jede Verzweigung führt zu zwei neuen kontextabhängigen Modellen und macht ein bisheriges Modell obsolet, entspricht also dem Hinzufügen eines Modells. Für null zusätzliche Modelle werden dem Entscheidungsbaum keine Kontextfragen hinzugefügt, es handelt sich also um ein kontextunabhängiges System. In diesem Fall besteht das System aus 97 Modellen, welche sich aus 16 Primitiven mal 6 Zuständen

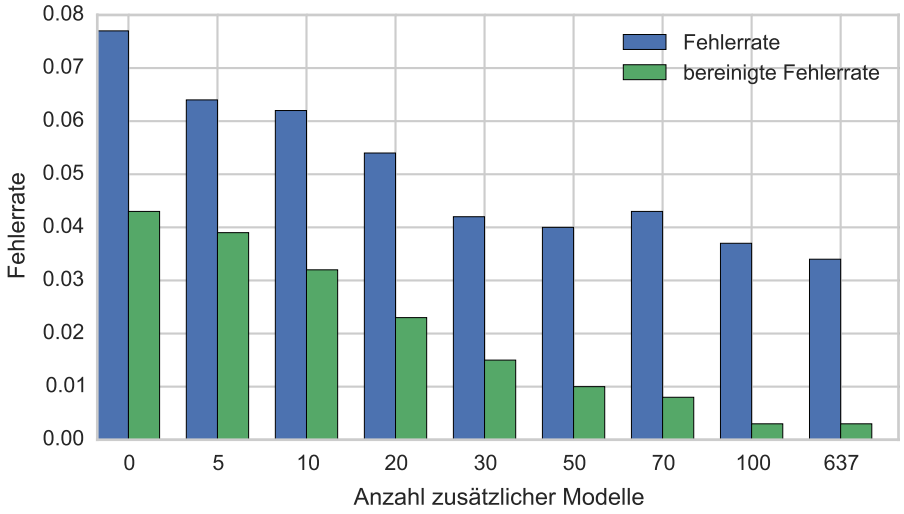


Abbildung 6.2 – Ergebnisse nach Anzahl der Triprimitive.

plus einen Zustand für das Ruhemodell ergeben. Für den Fall 637 zusätzlicher Modelle wird das Maximum an möglichen Teilungen erreicht, das heißt alle Triprimitive sind abgedeckt. Die maximale Anzahl an Modellen beträgt damit 734. Werte zwischen 100 und 637 zusätzlichen Modellen sind nicht angegeben, die Ergebnisse unterscheiden sich in diesem Bereich nicht signifikant.

Als alternatives Abbruchkriterium wurde auch die minimale Anzahl an Trainingsbeispielen getestet. Im vorliegenden Fall stellt die Menge der Trainingsdaten allerdings kein relevantes Problem dar, da auch für die maximale Anzahl von 734 Modellen die minimale Fehlerrate erreicht wird (siehe Grafik 6.2). Für jedes Modell sind genügend Trainingsdaten vorhanden. Da jedes Zeichen im Strich-Datensatz 15 mal wiederholt wurde, ist dies auch die Minimalanzahl der Trainingsbeispiele, die für jedes Triprimitiv vorliegt.

Die Fehlerraten zeigen, dass eine Modellierung auf Einzelstrichebene nutzbar ist und eine kontextabhängige Modellierung hier deutliche Vorteile bringt. Für den kontextunabhängigen Fall wird eine Fehlerrate von 7,7 %, für den kontextabhängigen Fall eine Fehlerrate von 3,3 % erreicht. Die bereinigten Fehlerraten betragen 4,3 % für den kontextunabhängigen Fall und 0,3 % für den kontextabhängigen Fall. Bereits wenige kontextabhängige Modelle führen zu einer deutlichen Verbesserung der Fehlerrate. Bei 20 zusätzlichen Modellen ergibt sich beispielsweise eine Reduktion der bereinigten Fehlerrate um 47 %. Ab ungefähr 100 zusätzlichen Modellen wird das Optimum in der bereinigten

Fehlerrate erreicht. Für das beste System kann eine relative Verbesserung der bereinigten Fehlerrate von 93 % gegenüber dem kontextabhängigen Fall erreicht werden.

Bei eine Modellierung auf der Ebene einzelner Striche sollte also in jedem Fall eine kontextabhängige Modellierung der Primitive gewählt werden, da diese einer kontextunabhängigen bei Weitem überlegen ist. Die optimale Anzahl von Modellen hängt von der Menge der zu modellierenden Zeichen und der verfügbaren Anzahl an Trainingsdaten ab. Die im Rahmen dieses Experimentes ermittelte Zahl kann dementsprechend nicht auf andere Zeichenmengen verallgemeinert werden.

6.4.2 Domänentransfer

Das Experiment zum Domänentransfer wurde auf dem Strich Datensatz durchgeführt. Ein Domänentransfer bedeutet in diesem Fall, dass die Zeichen in der Trainings- und Testmenge keine oder nur minimale Überdeckung haben. In den bisherigen Experimenten war die Anzahl und Art der Zeichen für die Trainings- und Testmenge immer identisch. Im Idealfall wäre es möglich, mit einer kleinen Menge von primitiven Modellen eine sehr große Menge von Schriftzeichen und Symbolen erkennen zu können, ohne über Trainingsdaten für diese Zeichen zu verfügen.

Es wurden verschiedene Kombinationen aus Trainings- und Testmenge evaluiert:

1. Training auf Primitiven (Strich.P), Test auf Buchstaben (Strich.B). Diese Kombination untersucht die Frage, wie gut ein Zeichensystem erkannt werden kann, wenn die Modelle ausschließlich auf isolierten Aufnahmen primitiver Striche trainiert wurden.
2. Training auf Primitiven, Kombinationen, Symbolen und Buchstaben, Test auf Zahlen. Mit dieser Variante wird die Frage untersucht, wie gut eine neue Zeichenmenge erkannt werden kann, wenn die Modelle bereits auf einer größeren Menge an Zeichen trainiert wurden.
3. Training auf Primitiven, Kombinationen und Buchstaben, Test auf Symbolen und Zahlen. Im Vergleich zum vorherigen Experiment wird die Trainingsmenge verkleinert und die Testmenge durch Hinzufügen der Symbole verdoppelt.

Die Ergebnisse für die drei Fälle sind in Tabelle 6.3 zusammengefasst. Neben der Fehlerrate wird für die zwei Fälle, bei denen auf Zahlen getestet wur-

Training	Test	Kontextunabh.	Kontextabh.
Strich.P	Strich.B	24,9	-
Strich ohne Strich.Z	Strich.Z	10,6 (0,7)	10,0 (0,0)
Strich ohne Strich.S,Z	Strich.S,Strich.Z	6,7 (1,4)	8,8 (3,5)

Tabelle 6.3 – Ergebnisse der Experimente zum Domänentransfer. Die in Klammern angegebenen Werte geben die bereinigte Fehlerrate, für welche Verwechslungen zwischen (0,6) nicht gezählt werden, an. Wird ausschließlich auf den Primitiven in Strich.P trainiert, entspricht ein kontextabhängiges System einem kontextunabhängigen, da alle Primitive ausschließlich im Kontext $(Ruhe|Ruhe)$ vorkommen.

de, auch die bereinigte Fehlerrate angegeben. Für diesen Wert wurden die Verwechslungen zwischen dem Paar (0,6) nicht als Fehler gezählt, da diese beide Zeichen durch die gleiche Strichsequenz modelliert werden. Ein Vergleich zwischen den Systemen kann nur auf Basis der bereinigten Fehlerrate vorgenommen werden. Für den Fall des Trainings auf Primitiven und des Tests auf Buchstaben unterscheidet sich die kontextabhängige nicht von einer kontextunabhängigen Modellierung, da in der Menge der Primitive jedes Primitiv genau einmal im Kontext $(Ruhe|Ruhe)$ vorkommt.

Wird nur auf den Primitiven trainiert, ist die Fehlerrate der Buchstabenerkennung mit knapp 25 % sehr hoch. Außer dem Buchstaben I, der als Triprimitiv $G_u(Ruhe|Ruhe)$ modelliert wird, kommt keines der Triprimitive der Trainingsdaten in den Testdaten vor. Dies bedingt die hohe Fehlerrate. Ausgehend von isolierten Aufnahmen der Primitive lassen sich also beliebige Zeichen nicht robust erkennen. Das Ergebnis ist aufgrund der Kontextabhängigkeit der Primitive so zu erwarten.

Erhöht man aber die Anzahl der vorkommenden Triprimitive in der Trainingsmenge, ist eine Erkennung unbekannter Zeichen sehr wohl möglich. Dies zeigen die zwei weiteren Experimente. Als Trainingsmenge wurden immer die Datensätze Strich.P, Strich.K und Strich.B verwendet. Der Datensatz Strich.P beinhaltet alle Striche aus der in Abschnitt 6.1.1 definierten Menge atomarer Striche. Damit ist sichergestellt, dass für jeden atomaren Strich Trainingsdaten zur Verfügung stehen. Der Datensatz Strich.K enthält zusätzlich alle geraden Striche noch in einer direkten Hintereinanderausführung eines Striches und seines im Sinne der Bewegungsrichtung direkten Gegenübers. Einem Strich nach rechts folgt also direkt ohne Pause noch ein Strich nach links zurück zum Ausgangspunkt. Für die kreisförmigen Bewegungen wurde auf diese Variante verzichtet, da dieser Fall in keiner der in dieser Arbeit untersuchten

Zeichenmengen auftritt. Die Buchstaben im Datensatz Strich.B wurden alle in der gleichen Reihenfolge der Striche ausgeführt. Für das erste Experiment wurde zusätzlich der Datensatz Strich.S für das Training verwendet.

Als Testmengen wurden für das erste Experiment der Datensatz Strich.Z und für das zweite Experiment die Datensätze Strich.Z und Strich.S verwendet. Der Datensatz Strich.Z enthält Überdeckungen mit der Trainingsdomäne. Die Null ist in der Bewegung identisch mit dem Buchstaben O. Die Zahl Sechs hat zwar keine identische Bewegung, wird aber durch die gleiche Strichsequenz modelliert. Ähnlich wie bei den Buchstaben P und D ist hier auf Basis der Strichmodellierung keine Unterscheidung möglich. Die Evaluation erfolgte sowohl auf allen zehn Ziffern, als auch auf der um die Null und Sechs reduzierten Menge. Die in Strich.S enthaltenen Symbole sind so gewählt, dass sie keine Überdeckung mit der Trainingsdomäne aufweisen.

Eine Erkennung der Zahlen funktioniert sowohl für den kontextunabhängigen als auch für den kontextabhängigen Fall sehr gut. Grundsätzlich ist es also möglich, mithilfe der Modellierung auf Einzelstrichebene auch unbekannte Zeichen robust zu erkennen.

Die Ergebnisse zeigen für die aus Zahlen und Symbolen bestehende Testmenge eine erhöhte bereinigte Fehlerrate für den kontextabhängigen Fall. Hier zeigt sich ein Problem der kontextabhängigen Modellierung, welches bei unterschiedlichen Trainings- und Testdomänen auftritt. Die meisten Fehler, abgesehen von dem Paar (0,6), sind Verwechslungen zwischen der Zahl 2 und dem Symbol des vertikal gespiegelten S. Dabei wird das gespiegelte S immer als Zahl 2 erkannt. Die Primitive K_{lu} und K_{ru}, aus denen das gespiegelte S zusammengesetzt ist, kommen nie in diesem Kontext in den Trainingsdaten vor. Das Primitiv K_{ru} kommt aber in den Buchstaben R, B, P, D und S in anderen Kontexten vor, die aber alle aufgrund der sich unterscheidenden Bewegungsrichtungen zu unterschiedlichen Signalen als im gespiegelten S führen. Das Ergebnis sind entsprechend spezifische kontextabhängige Modelle für das Primitiv K_{ru}, die zu einer schlechten Generalisierbarkeit in ungesehenen Kontexten führen. Die Zahl 2 wird als Sequenz K_{or}, G_{ul}, G_r modelliert. Das Primitiv K_{ul} kommt in den Trainingsdaten im Kontext einer Bewegung von rechts vor und ist damit in einigen Fällen anscheinend besser geeignet, um die Signale in der Mitte der Bewegung des gespiegelten S zu modellieren.

An diesem Beispiel wird das typische Problem einer kontextabhängigen Modellierung bei einer von der Trainingsdomäne unterschiedlichen Zieldomäne deutlich. Die kontextabhängige Modellierung führt zu spezifischeren Modellen der in der Trainingsdomäne vorkommenden Triprimitive. Beinhaltet die Testmenge nun Triprimitive, die in den Trainingsdaten nicht vorkommen, ge-

neralisieren die spezifischen kontextabhängigen Modelle schlecht. Hier kann also eine kontextunabhängige Modellierung trotz der bereits diskutierten Schwächen von Vorteil sein. Eine kontextabhängige Modellierung kann nur Vorteile bringen, wenn die Kontexte der Trainingsmenge auch in der Testmenge vorkommen.

Es bestehen mehrere Möglichkeiten, eine weitere Verbesserung der Erkennungsleistung im Falle eines Domänentransfers zu erreichen. Falls die Trainingsmenge verändert werden kann, können Triprimitive hinzugefügt werden, die häufig in den Testdaten vorkommen, um somit auch für die kontextabhängige Modellierung ein gutes Ergebnis zu erreichen. Falls Trainingsdaten für die Zieldomäne zur Verfügung stehen, aber keine Annotierung auf Ebene der Primitive vorhanden ist, kann mittels eines kontextunabhängigen Systems eine automatische Annotierung vorgenommen werden. Anschließend könnte entsprechend ein kontextabhängiges System auf der Zieldomäne trainiert werden. Diese beiden Varianten der Verbesserung werden in der vorliegenden Arbeit nicht evaluiert und könnten Gegenstand weiterer Forschung sein, insbesondere in Bezug auf die Erkennung von Zeichen anderer Schriftsysteme, wie beispielsweise dem Chinesischen.

6.4.3 Buchstabenerkennung

In diesem Experiment wird die Modellierung auf Strichebene mit der Modellierung auf Buchstabenebene verglichen. Dafür wird die personenunabhängige Fehlerrate für eine Strichmodellierung auf dem Datensatz Handschuh.B bestimmt. Als Vergleich dienen die Ergebnisse der Buchstabenerkennung aus Abschnitt 5.5.1 (basierend auf [AGS10]).

In diesem Experiment ist zu beachten, dass beide Modellierungsarten Vorteile haben. Eine buchstabenbasierte Modellierung erlaubt ein genau an den Buchstaben angepasstes Modell. Im Gegensatz zur einzelstrichbasierten Modellierung, in der Buchstaben auf eine Folge von abstrahierten Basisstrichen abgebildet werden, kann ein Buchstabenmodell exakt die spezifische Schreibbewegung des Buchstabens repräsentieren. Die einzelstrichbasierte Modellierung hingegen erlaubt, neben den anderen in Abschnitt 6.1 genannten Vorteilen, eine einfache Modellierung von Schreibvarianten. So können für einen Buchstaben im Wörterbuch beliebige Varianten eingetragen werden. Eine Modellierung von Schreibvarianten ist zwar grundsätzlich auch für Buchstabenmodelle möglich, erfordert aber entweder eine manuelle Annotation der Varianten in den Trainingsdaten oder ein datengetriebenes, unüberwachtes Clustering der aufgenommenen Daten in unterschiedliche Varianten. Infor-

Buchstaben	Striche
18,1 %	18,3 %

Tabelle 6.4 – Ergebnisse der personenunabhängigen Buchstabenerkennung für die buchstabenbasierte und die einzelstrichbasierte Modellierung.

melle Untersuchungen haben allerdings ergeben, dass ein hierarchisches Clustering basierend auf der *Dynamic-Time-Warping* Distanz der Signale keine natürliche Zuordnung in Varianten liefert. Für das hier beschriebene Experiment wurden alle Schreibvarianten in das Wörterbuch aufgenommen, die während der Aufnahme des Handschuh.B-Datensatzes beobachtet wurden.

Das kontextabhängige System, basierend auf Einzelstrichen, wird zuerst auf dem Strich-Datensatz erstellt und trainiert. Anschließend wird das System mittels einer Kreuzvalidierung über die einzelnen Personen evaluiert. Die Fehlerraten sind in Tabelle 6.4 angegeben und für beide Modellierungsarten vergleichbar. Die Vor- und Nachteile der jeweiligen Modellierungsarten gleichen sich offensichtlich aus. Die Modellierung auf Einzelstrichbasis ist allerdings deutlich flexibler. Sie erlaubt beispielsweise das einfache Hinzufügen weiterer Varianten. Auch eine Überführung in eine buchstabenbasierte Modellierung ist möglich. Die Gruppierung in Varianten ergibt sich dann auf natürliche Weise direkt aus der strichbasierten Modellierung.

6.5 Zusammenfassung

In diesem Kapitel wird eine Modellierung der Schrift auf Basis von einzelnen Strichen eingeführt. Es wird eine Menge von 16 Grundstrichen, den Primitiven, definiert, aus denen Buchstaben, Symbole und Zahlen zusammengesetzt werden. Weiter wird eine kontextabhängige Modellierung der Primitive eingeführt, da insbesondere das Beschleunigungssignal eines Primitives in starkem Maße von Vorgänger- und Nachfolgerprimitiv abhängt. In mehreren Experimenten werden die Vorteile einer Modellierung auf Strichebene nachgewiesen. In einem ersten Experiment wird gezeigt, dass eine Erkennung auf Basis der Einzelstrichmodelle mit niedrigen Fehlerraten möglich ist und dass eine Modellierung mit kontextabhängigen Modellen die Fehlerrate deutlich reduziert. Weiter wird gezeigt, dass auch ein Erkennung von Zeichen, die in den Trainingsdaten nicht vorkommen, möglich ist. Dies erfolgt am Beispiel der Erkennung der alphanumerischen Zahlen und einer Reihe von Symbolen. In Zukunft könnte dieser Ansatz aber auch die Übertragung der Erkennug

auf andere Schrift- oder Zeichensysteme ohne eine aufwändige vollständige Trainingsdatensammlung ermöglichen. Abschließend wird gezeigt, dass die Modellierung auf Einzelstrichebene für die personenunabhängige Buchstabenerkennung vergleichbare Fehlerraten zur Einzelbuchstabenmodellierung erreicht und damit aufgrund ihrer zusätzlichen Vorteile einer Modellierung auf Buchstabenebene vorzuziehen ist.

Kapitel 7

Das *Airwriting*-System

In diesem Kapitel wird der entwickelte Demonstrator des Airwriting-Systems zusammen mit Erweiterungen des Basissystems für die praktische Nutzung beschrieben. Für den Demonstrator wurde die Schrifterkennung um eine Detektionskomponente erweitert, welche Schriftsegmente aus dem laufenden Datenstrom von Sensordaten extrahiert und an die Erkennungskomponente weiterleitet. Zusätzlich wurde, um die über Funk zu übertragende Datenmenge zu reduzieren, eine Merkmalsextraktion auf Basis einer linearen Signalapproximation entwickelt und auf den vorhandenen Datensätzen evaluiert.

Die bisher vorgestellten Methoden und Experimente zielten darauf ab, die Genauigkeit der Schrifterkennung zu evaluieren und zu verbessern. Um eine real nutzbare Texteingabe für *Wearable Computing*-Anwendungen zu erhalten, sind allerdings noch weitere Herausforderungen zu addressieren. In der Einleitung wurden in Abschnitt 1.4 Kriterien zur Beurteilung der Eignung einer Texteingabmethode für *Wearable Computer* benannt. Die Kriterien der *ständigen und unmittelbaren Verfügbarkeit* sowie die *schnelle Texteingabe* stellen Anforderungen an das System, die über die reine Erkennung von Schrift hinausgehen. Zudem stellen auch begrenzte Batterie-, Rechen- und Speicherkapazitäten in Mobilgeräten und Sensoren Herausforderungen dar. In diesem Kapitel wird eine Architektur für ein System, welches die genannten Kriterien erfüllt und energieeffizient arbeitet, vorgestellt.

Über eine automatische Schriftdetektion wird eine ständige und unmittelbare Verfügbarkeit möglich gemacht. Der Nutzer muss keine explizite Aktivierung des Gerätes und keine explizite Segmentierung der Daten vornehmen. Die Detektion wird mit einer in nahezu Echtzeit laufenden Schrifterkennung kombiniert, um ein vollständig nutzbares System zu erhalten. Dieses System wurde als Demonstrator implementiert und beispielsweise auf der CeBIT 2014 vorgestellt.

Um das Problem der Batterielaufzeiten zu adressieren, wird zudem ein Verfahren zur Kompression der Sensordaten vorgestellt. Durch die Kompression verringert sich die über Funk zu übertragende Datenmenge. Das Verfahren funktioniert unter Echtzeitbedingungen und führt nur zu einer geringen Erhöhung der Fehlerrate.

7.1 Systemkomponenten

In Abbildung 7.1 sind die Komponenten eines vollständigen Prototyp-Systems dargestellt. Dies sind der Inertialsensor, eine Komponente zur Detektion von Handschrift, eine Erkennungskomponente und die Ausgabe. Die Erkennungskomponente entspricht weitgehend dem in Kapitel 4 beschriebenen Basissystem mit einer Anpassung der Vorverarbeitung an die Anforderungen der Datenverarbeitung in Echtzeit.

Im Folgenden wird die Detektionskomponente beschrieben, welche im kontinuierlichen Sensordatenstrom Segmente identifiziert, die möglicherweise Handschrift enthalten. Nur diese Segmente werden dann an die Erkennungskomponente weitergeleitet. Danach werden Methoden der Vorverarbeitung beschrieben, um sowohl eine Verarbeitung in Echtzeit zu gewährleisten als auch die zu übertragende Datenrate zu senken.

7.2 Detektion

In der Detektionsphase wird eine binäre Klassifikation des Datenstroms in Segmente, die Handschrift enthalten, und Segmente, die keine Handschrift enthalten, durchgeführt. Die als potentielle Handschriftsegmente klassifizierten Signalabschnitte werden zur Erkennungsphase weitergereicht, alle anderen werden verworfen. Idealerweise führt der Detektionsalgorithmus zu wenig Latenz in der Verarbeitungskette und identifiziert zuverlässig alle Hand-

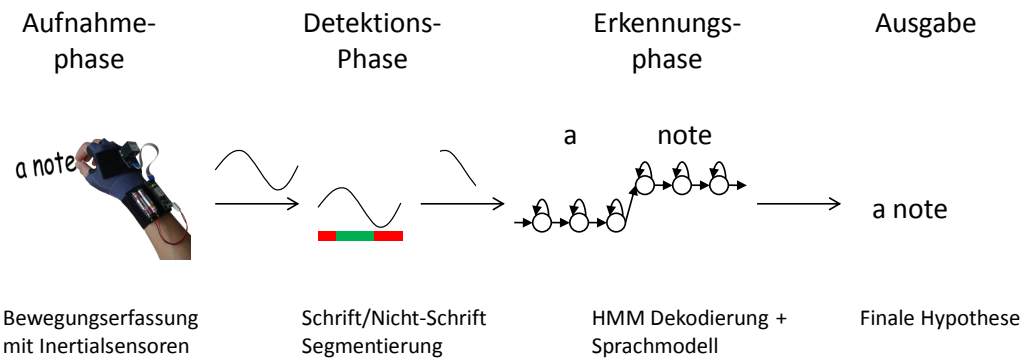


Abbildung 7.1 – Überblick über die Systemkomponenten des *Airwriting* Demonstrator-Systems.

schriftsegmente, ohne viele falsch-positiv Fehler zu produzieren. In Abbildung 7.2 wird das Problem anhand von Beispielsignalen verdeutlicht. Die dargestellten Beschleunigungs- und Winkelgeschwindigkeitssignale wurden über 19 min aufgenommen und beinhalten drei Handschriftsegmente. Die Aufgabe in der Detektionsphase ist die Trennung dieser Handschriftsegmente von zeitlich davor und dahinter liegenden Alltagsaktivitäten. Dies wird über eine Klassifikation überlappender Fenster mit einer Support-Vektor-Maschine (SVM) und anschließender Entscheidungsfusion der überlappenden Klassifikationsergebnisse realisiert. Das Verfahren wird in den folgenden Abschnitten erläutert.

7.2.1 Merkmalsextraktion

Die Daten werden über eine gleitende Fensterung in gleichlange, überlappende Fenster zerlegt. Auf jedem Fenster w werden die folgenden Merkmale berechnet, wobei N_w die Anzahl der Messwerte und $\boldsymbol{a}_w$ und $\boldsymbol{g}_w$ die Beschleunigungs- und Drehratenmesswerte im Fenster w bezeichnen:

- Mittlere Winkelgeschwindigkeit (g_{mittel})

$$\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{g}_w(i)\|_2$$

- Mittlere mittelwertbereinigte Beschleunigung ($a_{mittelsub}$)

$$\frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{a}_w(i) - \bar{\boldsymbol{a}}_w\|_2$$

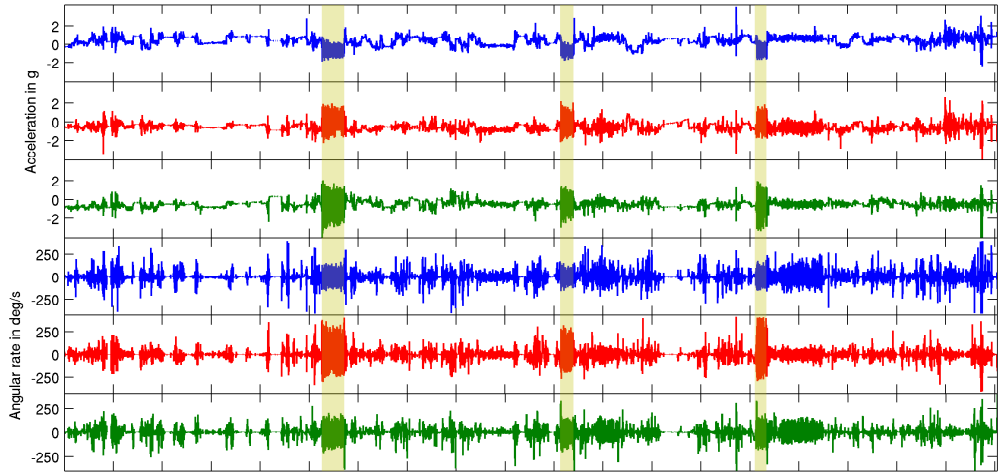


Abbildung 7.2 – Beispielsignale mit sporadischer Handschriftaktivität. Die drei farbig hinterlegten Segmente enthalten Handschrift, die restlichen Bereiche enthalten Alltagsaktivität.

- Energie pro Frequenz zwischen 0 Hz und 8 Hz in 8 Bändern von 1 Hz Breite ($energie$)

Die Merkmale wurden nach visueller Analyse der Signalstatistiken in Abbildung 7.4 gewählt. Handschriftbewegungen haben eine höhere Frequenz und Amplitude verglichen mit Nichthandschriftbewegungen. Im Gegensatz zu alltäglichen Bewegungen gibt es beispielsweise bei Handschrift eine erhöhte Energie im Bereich zwischen 2 und 5 Hz (siehe auch Abschnitt 2.1).

7.2.2 Klassifikation

Abbildung 7.3 zeigt die Funktionsweise des gesamten Detektionsalgorithmus. Im mittleren Teil der Abbildung 7.3 sind die überlappenden Fenster dargestellt, die Farbe gibt dabei das Klassifikationsergebnis an, grün steht für „als Handschrift klassifiziert“, rot für „nicht als Handschrift klassifiziert“. Der durchgehende Balken im unteren Teil der Abbildung 7.3 gibt das finale Klassifikationsergebnis nach der Fusionierung der Klassifikationsergebnisse auf den Fenstern an. Jeder Messpunkt ist Teil mehrerer überlappender Fenster. Für jedes Fenster w_t liefert die SVM $C_{\mathrm{SVM}}(w_t)$ den Wert 0 für die Klasse Handschrift und sonst den Wert 1 zurück. Jeder Messwert wird durch den kombinierten Klassifikator C_{COMB} klassifiziert. Ein Messwert $s(i)$ wird als

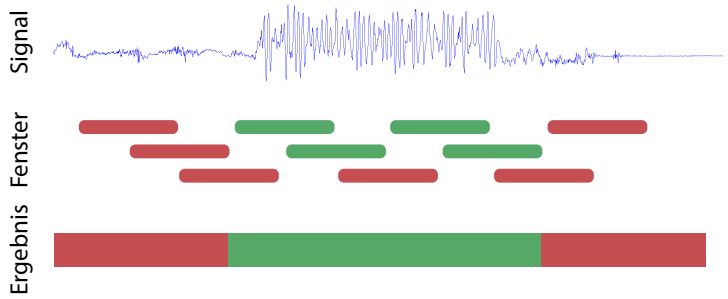


Abbildung 7.3 – Schematische Darstellung des Detektionsalgorithmus.

Handschrift klassifiziert, wenn mindestens ein Fenster, welches $s(i)$ enthält, als Handschrift klassifiziert wurde:

$$C_{\text{COMB}}(s(i)) = \max_{k:s(i)\in w_k} C_{\text{SVM}}(w_k) \tag{7.1}$$

Die Anzahl der überlappenden Fenster hängt von der gewählten Überlappung ab. Eine Sequenz von Messwerten, die als Handschrift klassifiziert wurden, wird als Handschriftsegment bezeichnet. Im Gegensatz zu anderen Entscheidungsfusionsverfahren, wie beispielsweise einer Mehrheitsentscheidung, favorisiert das gewählte Verfahren die Erkennung von Schrift statt keiner Schrift, ist also auf eine hohe Trefferquote optimiert. Dadurch werden eher zu viele Daten als Handschriftsegment klassifiziert und an die Erkennung weitergereicht als zu wenige. Dies ist ein gewolltes Verhalten, da fälschlicherweise als Schrift segmentierte Daten in der Erkennungsphase noch aussortiert werden können, aber fälschlicherweise als Nichtschrift segmentierte Daten gar nicht an die Erkennung durchgereicht werden und damit in der Regel zu Fehlern führen.

Sobald ein Messpunkt eindeutig als Handschrift klassifiziert ist, wird er an die Erkennungsphase weitergereicht, damit entspricht die durch die Detektion entstehende Latenz der Länge eines Fensters.

Für falsch-positiv erkannten Messpunkte gibt es zwei Fehlertypen. Entweder die Messpunkte liegen am Anfang oder Ende eines tatsächlichen Schriftsegmentes und verwischen damit die Grenzen des Segments oder sie bilden ein eigenes Segment, welches nicht mit tatsächlicher Handschrift korrespondiert.

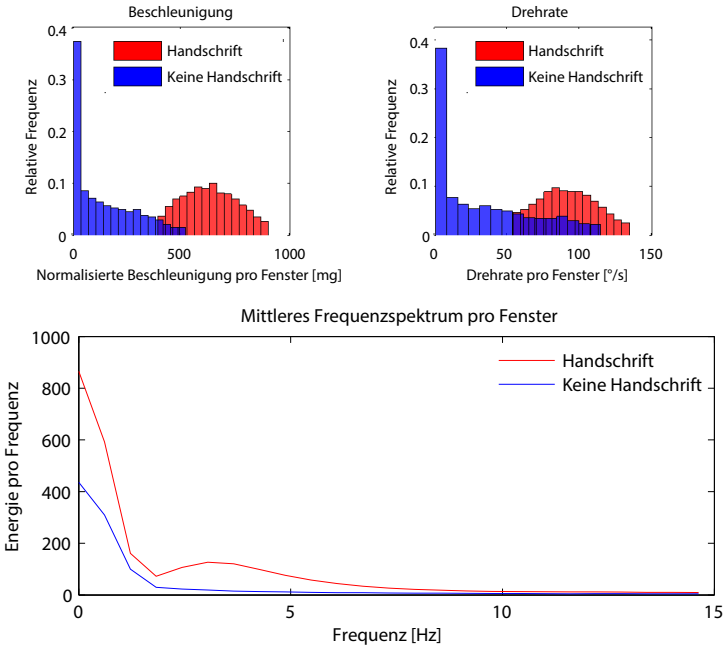


Abbildung 7.4 – Vergleich der Merkmale g_{mittel}, $a_{mittelsub}$ und $energie$ auf Aufnahmen mit (rot) und ohne (blau) Handschrift. Beschleunigung und Winkelgeschwindigkeit sind unterschiedlich verteilt. Die Frequenzverteilung unterscheidet sich ebenso.

Der erste Fall stellt dann ein Problem dar, wenn die Segmentgrenzen um viele Messwerte, beispielsweise äquivalent zur Länge eines Buchstabens, verschoben sind. Dies tritt auf, wenn der Nutzer fast nahtlos von einer schriftähnlichen Bewegung, wie etwa Winken, in echtes Schreiben oder umgekehrt übergeht. Der zweite Fall führt dann zu einem Fehler, wenn der Erkenner eine vom leeren Wort unterschiedliche Hypothese ausgibt. Die falsch detektierten Segmente sind meist allerdings sehr kurz und führen entsprechend nur zu sehr kurzen Hypothesen (0-3 Buchstaben). Über eine Mindestlänge N der Hypothese in Buchstaben wird hier eine weitere Filterung in der Erkennungsphase vorgenommen. Ist der Text kürzer als diese Mindestlänge, wird er verworfen.

7.3 Vorverarbeitung der Erkennung

Für die Echtzeiterkennung muss die in Abschnitt 4.2 beschriebene Vorverarbeitung der Erkennung leicht angepasst werden, da sie in der beschriebenen

Form voraussetzt, dass zum Zeitpunkt der Berechnung der Merkmale bereits das vollständige Schriftsegment zur Verfügung steht. Um eine minimale Latenz in der Erkennung zu erhalten, ist es allerdings wünschenswert, dass die Dekodierung bereits kurz nach dem Erhalt der ersten Daten beginnen kann. Dies ermöglicht es außerdem, dem Nutzer eine Rückmeldung über den Erkennungsprozess bereitzustellen. Zudem wird an dieser Stelle eine weitere Vorverarbeitungsvariante vorgestellt, die zu einer deutlichen Datenkompression führt und ebenfalls mit geringer Latenz auf kontinuierlichen Daten berechnet werden kann.

7.3.1 Laufende z-Normalisierung

Für die Experimente in Kapitel 5 werden die Normalisierungsschritte in der Merkmalsextraktion immer auf dem gesamten verarbeiteten Buchstaben, Wort oder Satz berechnet. Im Falle einer kontinuierlichen Erkennung ist dies nicht möglich, da ansonsten bis zum Ende der Eingabe mit dem Start der HMM-Berechnung gewartet werden muss. Die Normalisierung wird daher kontinuierlich berechnet. Dadurch entsteht eine Abweichung zur regulären Normalisierung, die mit der Zeit kleiner wird. Da die Detektionskomponente allerdings immer auch einen Teil des Sensorsignals vor der eigentlichen Schriftbewegung mit erfasst, hat dieser Effekt keinen gravierenden Einfluss. Die Erfahrungen mit dem Demonstrator zeigen außerdem, dass die Erkennung darunter nicht wesentlich leidet. Eine formale Evaluierung des Effekts wurde nicht durchgeführt.

7.3.2 SWAB

Die Grundlage für die komprimierte Merkmalsextraktion ist der SWAB-Algorithmus von Keogh et al. [KCHP01]. Der SWAB-Algorithmus berechnet eine stückweise, lineare Approximation eines eindimensionalen Signals, das heißt das Signal wird anhand einer Folge von verbundenen Geradensegmenten dargestellt. Die Darstellung ist daher durch die Angabe der Punkte an den Segmentgrenzen eindeutig definiert. Die Berechnung verbindet ein gleitendes Fenster (engl. sliding window) mit einer darin ausgeführten bottom-up basierten Berechnung der Approximation. Dabei wird innerhalb eines Fensters von der feinst möglichen Approximation aus gestartet und die Segmente werden solange zu größeren Segmenten vereinigt, bis eine definierte Fehlerschwelle überschritten ist. Die bottom-up Approximation gewährleistet eine

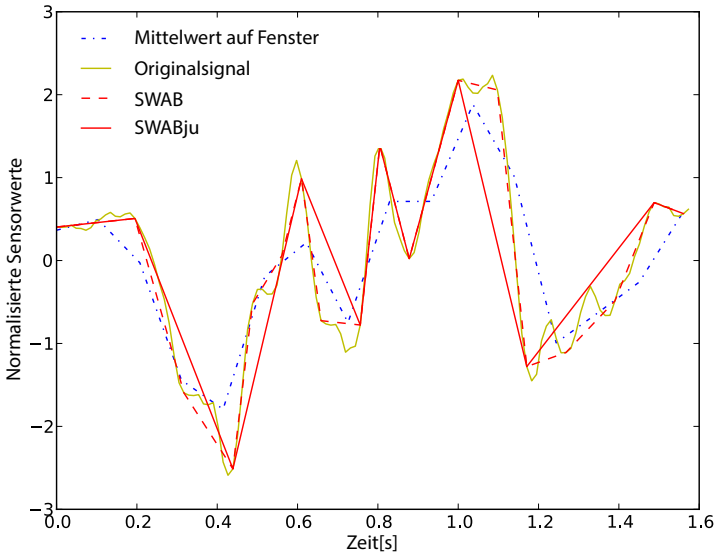


Abbildung 7.5 – Ergebnisse des SWAB und SWABju Algorithmus für ein Originalsensorsignal im Vergleich zu einer fensterbasierten Mittelwertbildung.

hohe Qualität der Approximation, während die Fensterung eine Verarbeitung von kontinuierlich eingehenden Daten erlaubt. Die Speicher- und Laufzeitkomplexität des Algorithmus ist linear und er kann einfach auf kleinen Sensorknoten, beziehungsweise Mikroprozessoren, implementiert und ausgeführt werden. Juncker et al. schlagen in [JALT08] eine Erweiterung des Algorithmus vor, um die Anzahl der Segmente und damit der Daten weiter zu reduzieren. Der Algorithmus wird im Weiteren als SWABju bezeichnet. Benachbarte Segmente werden abhängig von der Ähnlichkeit ihrer Steigung vereinigt. Anhand der Wahl des maximalen Winkels, bis zu dem Segmente zusammengefasst werden, kann eine Feineinstellung des Algorithmus erfolgen. Die Resultate der Approximation durch die zwei Algorithmen sind in Abbildung 7.5 dargestellt.

7.3.3 SWABmd

Eine Erweiterung des eindimensionalen SWAB-Algorithmus auf den multidimensionalen Fall ergibt sich nicht auf natürliche Weise. Eine Möglichkeit ist es, den Fehler, den eine Segmentierung erzeugt, über alle Dimensionen zu berechnen. Sind die Dimensionen allerdings unabhängig, was in unserem Fall zumindest teilweise gegeben ist, führt dies zu einer hohen Anzahl an Seg-

menten. Die damit erreichten Segmentierungen waren für den vorliegenden Anwendungsfall unbefriedigend. Die in dieser Arbeit gewählte Erweiterung führt zuerst eine Approximation für jede Dimension getrennt durch und vereinigt anschließend die Segmentgrenzen. Dieser multidimensionale SWAB-Algorithmus wird hier als SWABmd bezeichnet. Zuerst wird für jede Signaldimension unabhängig eine lineare stückweise Approximation mit dem SWABju Algorithmus berechnet. Die Zeitpunkte aller Segmentgrenzen aller Dimensionen werden anschließend vereinigt, um eine stückweise lineare Approximation des Gesamtsignals zu erhalten. Für jede Segmentgrenze werden die Daten der anderen Dimensionen jeweils aus den Approximationen interpoliert. Abbildung 7.6 illustriert die Arbeitsweise des Algorithmus. Durch die Interpolation der Sensorwerte in den anderen Dimensionen ist eine Segmentgrenze durch den Zeitpunkt, den Sensorwert und die Dimension, in der die Segmentgrenze auftrat, zu übertragen. Jede Segmentgrenze kann entsprechend als Tripel $(\Delta t, y, i)$ dargestellt werden, wobei Δt die Zeit in Form der Anzahl Abtastwerte seit dem letzten Segment enthält, y den Sensorwert in der Dimension i, in der die Segmentgrenze berechnet wurde.

Für die Berechnung der Kompressionsrate wird davon ausgegangen, dass Δt und d zusammen die gleiche Anzahl Bits benötigen wie die Sensorwerte selbst, die mit 16 Bits übertragen werden. Im Fall von sechs Dimensionen werden entsprechend 3 Bit für den Wert von i und 13 Bit für Δt reserviert. Der maximale Zeitabstand zwischen zwei Segmenten beträgt damit bei einer Abtastfrequenz von 82 Hz ungefähr 100 ms und ist damit mehr als ausreichend dimensioniert. Damit ergibt sich eine Größe von 32 Bit pro Tripel. Auf der Empfängerseite muss für die Durchführung der Interpolation gewartet werden, bis in allen anderen Dimensionen ebenfalls eine Segmentgrenze empfangen wurde. Dies führt zu einer Verzögerung, die abhängig von der Frequenz des Sensorsignals ist. Im Falle von Schriftbewegungen findet in allen Dimensionen genügend Aktivität statt, so dass diese Verzögerung, die ungefähr im Bereich 0 s bis 0,5 s liegt, kein Problem darstellt.

7.4 Evaluation

7.4.1 Detektion

Zum Training der SVM wurden die Trainingsdaten aus dem Detektion-Datensatz sowie der vollständige Handschuh-Datensatz verwendet. Für die Evaluation der Detektionsphase wurden die Testdaten aus dem Detektion Datensatz

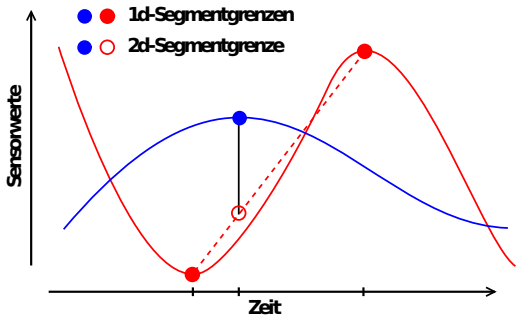


Abbildung 7.6 – Illustration der Arbeitsweise des SWABmd Algorithmus. Die 1d-Segmentgrenzen werden mit dem SWABju Algorithmus berechnet und dann vereinigt. Die Markierungen auf der Zeitachse geben die vereinigten Segmentgrenzen aller Dimensionen an. Für eine Segmentgrenze (blauer Punkt) werden die Werte der jeweils anderen Dimensionen auf deren linearer Approximation (rotes Geradensegment zwischen den roten Punkten) interpoliert (roter Kreis).

verwendet. Die benutzte Fenstergröße wurde zwischen 400 und 1400 Messpunkten, die Fensterüberlappung zwischen 0 und 700 Messpunkten variiert. Abbildung 7.7 zeigt die Ergebnisse der Evaluation anhand Genauigkeit, Trefferquote, Spezifität und F-Maß.

Die Trefferquote ist mit 97 % bis 99.9 % durchgängig sehr hoch, wobei erwartungsgemäß eine große Überlappung der Fenster bedingt durch die Entscheidungsfusion die höchste Trefferquote erzielt. Ein Nachteil einer großen Überlappung ist der erhöhte Rechenaufwand. Die Genauigkeit und Spezifität profitiert wiederum von kurzen Fenstern mit geringer Überlappung. Die Maxima betragen hier für die Trefferquote 35 % und die Spezifität 92 %, welche beide für eine Fensterlänge von 0,61 s (500 Abtastwerte) und 50 % Überlappung erreicht werden. Da eine hohe Trefferquote für die gegebene Anwendung Priorität hat wird eine Fensterbreite von 0,85 s (700 Abtastwerte) und eine Überlappung von 0,17 s (140 Abtastwerte) gewählt. Für diese Werte wird eine Trefferquote von 99 % und eine Genauigkeit von 26 % erreicht.

Zusätzlich wurde der Einfluss der verschiedenen Merkmale auf die Detektionsleistung untersucht. Die Ergebnisse sind in Abbildung 7.8 dargestellt. Die mittlere Amplitude hat den höchsten Informationsgehalt, verglichen mit den anderen Merkmalen. Allerdings liefert die Kombination aller Merkmale die höchste Detektionsleistung.

Die mittlere Dauer der falsch-positiv Segmente ist 1,58 s (σ = 1,64 s). Die mittlere Dauer eines Buchstabens im Datensatz Handschuh.S beträgt 0,83 s

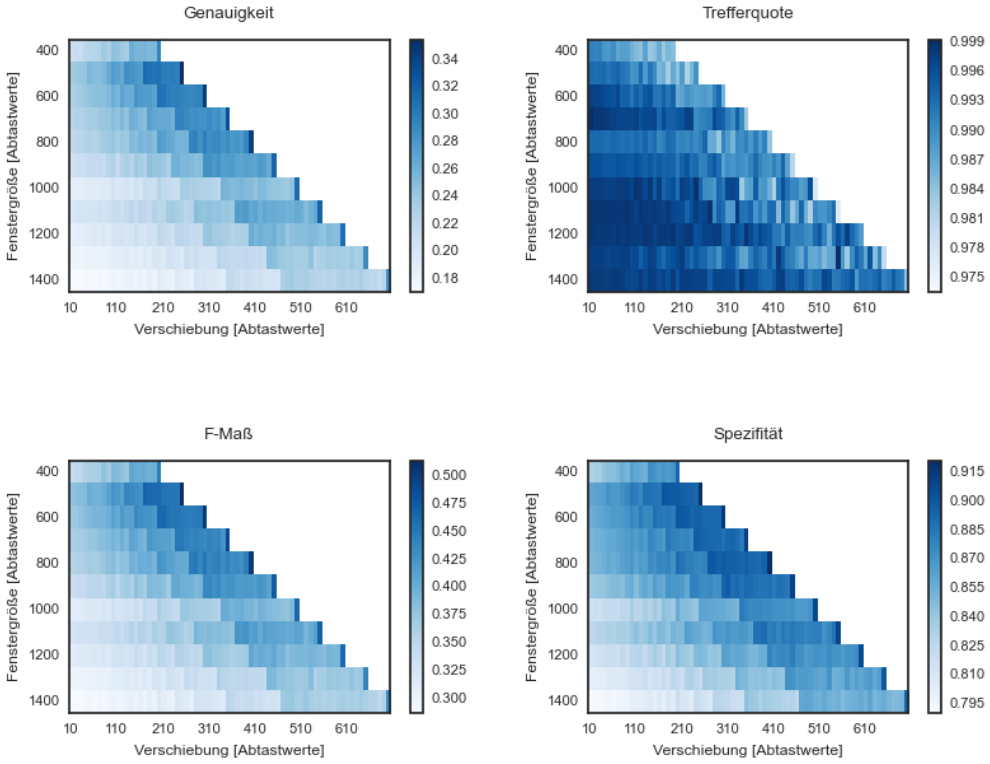


Abbildung 7.7 – Genauigkeit, Trefferquote, F-Maß und Spezifität der Detektion für unterschiedliche Fensterlängen und Überlappungen.

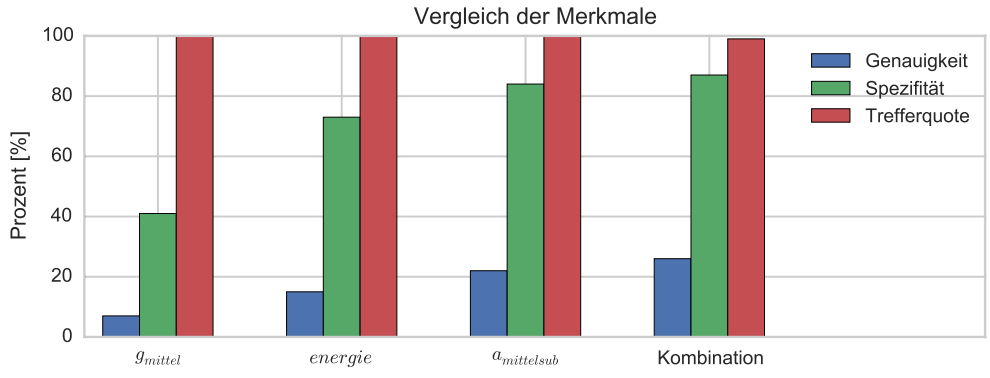


Abbildung 7.8 – Vergleich der Ergebnisse der Detektion für die verschiedenen Merkmale einzeln und in Kombination.

N	0	1	2	3
	91.3 %	98.2 %	99.3 %	99.5 %

Tabelle 7.1 – Prozentualer Anteil verworfener falsch-positiv Segmente für verschiedene Werte der Hypothesenlänge N in Buchstaben.

($\sigma = 0{,}17\,\mathrm{s}$). Die Dauer der falsch-positiv Segmente entspricht also in etwa der Schreibdauer von zwei Buchstaben. Entsprechend wurde die Filterung in der Erkennungsphase für die maximale Anzahl von 0, 1, 2 und 3 Buchstaben in der Hypothese ausgewertet. Tabelle 7.1 gibt den Anteil korrekt verworfener falsch-positiv Segmente an. Unter der Annahme, dass der Nutzer keine einzelnen sehr kurzen Wörter eingeben will, kann mit diesem Verfahren ein Großteil der falsch-positiv Segmente aus der Detektionsphase in der Erkennungsphase verworfen werden. Für die Eingabe von Text gilt diese Annahme im Regelfall. Soll aber beispielsweise über die Eingabe von textuellen Kürzeln mit einem oder wenigen Buchstaben eine Steuerung verschiedener Funktionen erfolgen, wäre dieser Ansatz ungeeignet. Auf dieses Szenario wird in Kapitel 8 im Rahmen der Detektion und Erkennung von Gesten eingegangen.

7.4.2 Merkmalsextraktion mit SWABmd

Die mit dem SWABmd Algorithmus generierten Merkmale wurden hinsichtlich der damit erzielbaren Erkennungsleistung mit einer Mittelwertbildung auf einem gleitenden Fenster verglichen. Für die Evaluation wurde der Datensatz Handschuh.B verwendet. Die Modellierung entspricht der in Kapitel 5 vorgestellten Buchstabenmodellierung. Es wurden statische Topologien verwendet, wobei für höhere Kompressionsraten die Anzahl der Zustände nach Notwendigkeit verringert wurden, um sicherzustellen, dass mehr Merkmalsvektoren als HMM Zustände vorhanden sind. Es wurde eine Kreuzvalidierung über die Personen durchgeführt, um die personenunabhängige Fehlerrate zu bestimmen.

Die Ergebnisse sind in Abbildung 7.9 dargestellt. Die Fehlerraten sind in Abhängigkeit der Kompressionsrate dargestellt. Die Kompressionsrate berechnet sich, indem die Datenmenge nach Kompression durch die Originaldatenmenge geteilt wird. Für die gefensterte Mittelung kann über die Fenstergröße die Kompressionsrate frei definiert werden. Die Ergebnisse sind als unterbrochene, blaue Linie eingetragen. Abhängig vom Wert des Kriteriums zur Vereinigung der Segmente im SWABju-Algorithmus, werden für SWABmd Kom-

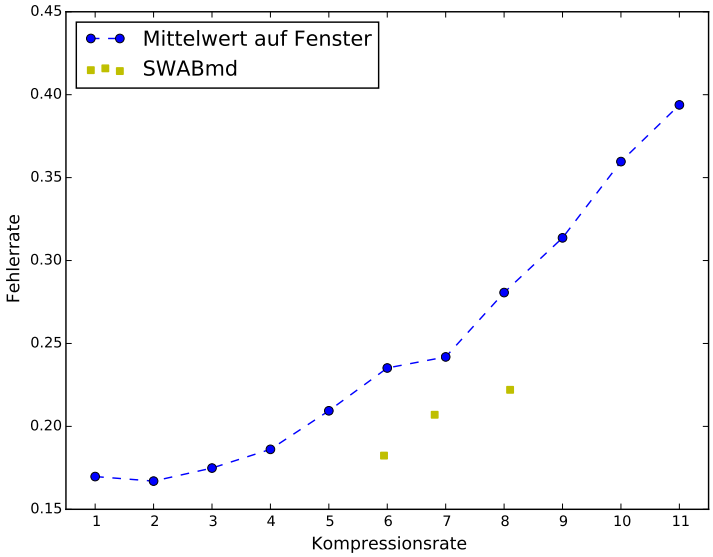


Abbildung 7.9 – Ergebnisse der Datenkompression mit SWABmd im Vergleich zu der gefensterten Mittelung.

pressionsraten von sechs bis acht erreicht. Diese sind als Punkte in der Grafik dargestellt. Bei einer Kompressionsrate von zwei erreicht das Vergleichssystem mit einer Fehlerrate von 16,7 % den besten Wert. Mit SWABmd wird für eine Kompressionsrate von 6 eine Fehlerrate von 18,2 % erreicht, im Vergleich zu 23,5 % für das Vergleichssystem. Dies entspricht einer relativen Reduktion der Fehlerrate um 22,5 %. In Anwendungsfällen, in denen die eine lange Batterielaufzeit nötig ist oder nur stark begrenzte Energieressourcen auf den Sensoren zur Verfügung stehen und daher die über Funk zu übermittelnden Daten minimiert werden müssen, kann die hier eingeführte Methode verwendet werden. Damit wird eine geringe Datenrate bei gleichzeitig nur wenig erhöhter Fehlerrate erreicht.

7.4.3 Evaluation des Gesamtsystems

Die im Rahmen des Detektion-Datensatzes gesammelten Testdaten können nicht nur für die Evaluierung der Detektionsphase genutzt werden, sondern auch zu Evaluierung des Gesamtsystems. Der Detektion Datensatz enthält große Teile unterschiedlicher Alltagsaktivitäten. Diese wurden in einem realistischen Alltagsszenario bei den Probanden zuhause aufgenommen und die Probanden wurden angehalten, während der Aufnahme unterschiedliche Ak-

tivitäten auszuführen. Es handelt sich dabei auch um bewegungsintensivere Aktivitäten als dies beispielsweise bei einer Bürotätigkeit der Fall ist. Das Schreiben erfolgte willkürlich darin eingebettet. Im Gegensatz zu den im Rahmen des Handschuh-Datensatzes im Labor aufgenommenen Daten wurde zudem in unterschiedlichen Größen und Handhaltungen geschrieben und es wurden keine Ruhephasen vor und nach dem Schreiben eines Satzes erzwungen. Entsprechend sind auch die Bewegungen kurz vor und nach dem Schreiben Teil des detektierten Schriftsegments. Der Detektion Datensatz beinhaltet 68 Wörter in 17 Sätzen. Aufgrund der Größe des Datensatzes ist die Evaluation darauf beschränkt, die grundsätzliche Funktionsweise des Gesamtsystems zu verifizieren und einen Eindruck der erreichbaren Erkennungsgenauigkeit zu erhalten.

Der für die Evaluation des Gesamstsystems genutzte Erkenner verwendet eine Buchstabenmodellierung gemäß Kapitel 5 und wurde sukzessive auf den Datensätzen Handschuh.B, Handschuh.W und Handschuh.S trainiert. Da die Probanden des Detektion Datensatzes bereits Daten zu den Handschuh Datensätzen beigesteuert haben, ist die Evaluation nicht personenunabhängig.

Für die Evaluation werden alle Daten des Detektion Datensatzes durch die in diesem Kapitel beschriebene Architektur aus Detektions- und Erkennungsphase geleitet. Der Parameter N für die Mindestlänge der Hypothesen (siehe 7.2.2) wird auf $N = 3$ gesetzt. Insgesamt blieben damit 19 Handschriftsegmente mit Hypothesen mit mehr als drei Buchstaben übrig. Auf diesen Segmenten wird eine Wortfehlerrate von 17 % erreicht. Werden die zwei falsch-positiven Handschriftsegmente von der Evaluierung ausgenommen, wird eine Wortfehlerrate von 7 % erreicht, da die falsch-positiv Segmente naturgemäß die Wortfehlerrate stark erhöhen. Im Vergleich zu den Ergebnissen aus der Airwriting Evaluation in Abschnitt 5.5.3 liegen die Ergebnisse damit im zu erwartenden Bereich. Insbesondere sollte bedacht werden, dass das Erkennungsproblem durch die Art der Datenaufnahme schwieriger als die Erkennung auf Labordaten ist.

7.5 Demonstrator

Basierend auf der in Kapitel 5 vorgestellten kontinuierlichen Satzerkennung wurde ein Demonstrator entwickelt. Dieser erlaubt die Detektion und Erkennung von Airwriting in nahezu Echtzeit. Das System nutzt das in Abschnitt 3.1 beschriebene Armband als Sensor.

Abbildung 7.10 – Präsentation des Demonstrators auf der CeBIT 2014.

Die Modellierung erfolgt auf Buchstabenebene und entspricht dem in Abschnitt 5.5.3 beschriebenen Ansatz zur personenunabhängigen Erkennung kontinuierlich geschriebener Sätze. Statt der Kreuzvalidierung auf dem Datensatz Handschuh.S wurde der gesamte Datensatz als Trainingsmenge verwendet. Die Ausgabe des Systems liefert in kurzen Abständen während des Schreibens die jeweils aktuell wahrscheinlichste Hypothese für die zu diesem Zeitpunkt bereits zur Verfügung stehenden Schriftdaten. Damit kann der Prozess der Erkennung visualisiert werden.

Es wurde bereits mehrfach auf Konferenzen [AS12] und Veranstaltungen sowie in mehreren Fernsehbeiträge (beispielsweise 3Sat nano, Pro7 Galileo, SWR Landesschau) vorgestellt und demonstriert. Im Rahmen der CeBIT 2014 wurde der Demonstrator auf dem KIT Stand ausgestellt und konnte von den Messebesuchern getestet werden. Abbildung 7.10 zeigt ein Foto der Demonstration am KIT Stand auf der CeBIT 2014 in Hannover. Darauf folgten weitere auch internationale Medienberichte (beispielsweise Foxnews, Dailymail, gizmag, Technology Review).

7.6 Zusammenfassung

In diesem Kapitel wurde ein Gesamtsystem zur Erkennung von Airwriting vorgestellt und evaluiert. Zusätzlich zum Erkennungssystem wurde eine Detektionskomponente eingeführt, welche eine binäre Klassifikation des Sensorstroms in Schrift- und Nichtschrift-Daten vornimmt. Dabei wird ein hybrider

Ansatz zur Segmentierung verwendet: Die Detektionskomponente ist darauf optimiert, eine hohe Trefferquote zu erzielen. Falsch-positiv Detektionen werden anschließend weiter über die Hypothesenlänge ausgefiltert. Neben der isolierten Evaluation der Detektionskomponente wird auch eine Evaluation des Gesamtsystems diskutiert. Auch unter realistischen Bedingungen erreicht die Erkennungsphase geringe Fehlerraten von 7 %, allerdings lassen fälschlicherweise erkannte Schriftsegmente die Fehlerrate auf 17 % steigen. Eine Möglichkeit, die Fehlerrate weiter zu reduzieren, wäre die Einbeziehung des Handlungskontextes des Nutzers.

Zusätzlich wurden in diesem Kapitel auch Techniken zur Reduktion der über Funk zu übertragenden Daten eingeführt und evaluiert. Damit können geringe Fehlerraten mit hohen Batterielaufzeiten kombiniert und somit eine Anwendung auf mobilen Geräten mit stark beschränkten Ressourcen ermöglicht werden. Hierzu wurde basierend auf dem eindimensionalen SWAB-Algorithmus eine mehrdimensionale Erweiterung entwickelt, welche eine stückweise lineare Approximation des Sensorsignals liefert. Bei einer Kompression des Originalsensorsignals um den Faktor sechs wird damit eine um 22,5 % niedrigere Fehlerrate als für das Vergleichsverfahren erreicht. Liegt der Fokus allerdings auf einer minimalen Fehlerrate, dann wird mit den unkomprimierten Daten ein besseres Ergebnis erreicht.

Kapitel 8

Das *Airactions*-System

In diesem Kapitel wird das Airactions-System vorgestellt. Es verbindet die Texteingabe via Airwriting mit einer Gestensteuerung. Durch diese Kombination entsteht eine freihändig nutzbare, mobile Gestenschnittstelle mit hohem Funktionsumfang. Die Wahl der Steuergesten wird motiviert und die Verfahren zur Gestendetektion und Erkennung werden eingeführt und evaluiert. Anhand der experimentellen Ergebnisse werden Konsequenzen für das Design solcher Schnittstellen diskutiert. Fabian Winnen hat ihm Rahmen seiner Bachelorarbeit die wesentlichen Methoden und Ergebnisse der Gestenerkennung und Gestensteuerung erarbeitet.

8.1 Überblick

Die Eingabe von Text stellt nur eine Komponente eines mobilen Interaktionssystems dar. Der Nutzer muss auch die Möglichkeit haben, einzelne Kommandos, wie beispielsweise die Navigation durch Menüs oder die Selektion von Einträgen, auszuführen. Dies kann beispielsweise über eine kleine Menge von Hand- und Armgesten zur Steuerung realisiert werden. Auch räumliche Kommandos, wie Verschiebungen oder eine Größenänderung, lassen sich besser über Gesten als über Text kommunizieren. Erst die Kombination von Steuergesten und Texteingabe erlaubt eine benutzerfreundliche, vollständig freihändige Bedienung einer mobilen gestenbasierten Schnittstelle.

In diesem Abschnitt werden die bisher entwickelten Methoden zur Schrifterkennung auf die Erkennung von Kommandogesten übertragen und evaluiert. Für die Gestendetektion werden Ansätze aus den Arbeiten von Park et al. [PLH+11], Prekopcsak [Pre08] und Raffa et al. [RLNS10] kombiniert. Das vollständige System wird unter realistischen Bedingungen evaluiert. Die Evaluation geht über die noch sehr eingeschränkten Evaluationen der genannten Arbeiten hinaus.

Basierend auf den Methoden zur Gestenerkennung und Detektion wird ein vollständig mobil einsetzbarer Demonstrator einer Benutzerschnittstelle, welche *Airwriting* und Kommandogesten integriert, implementiert. Diese Kombination von Steuergesten und Texteingabe wird im Folgenden als *Airactions* bezeichnet. Die Einsatzszenarien für derartige Systeme reichen von Endnutzer-Anwendungen, wie der einfachen Steuerung eines Musikspielers auf dem Smartphone, bis hin zur Bedienung von Augmented-Reality Anwendungen. Der Demonstrator implementiert die Benutzerschnittstelle für drei Instanzen des *Airactions*-Systems zur Bedienung (1) eines Musikspielers, (2) eines Telefons inklusive der Auswahl von Kontakten in einem Smartphone sowie (3) der Abarbeitung einer Checkliste für eine technische Inspektion.

Die Arbeiten zur Erkennung der Gesten basieren auf der Bachelorarbeit von Fabian Winnen [Win14]. Einige der Grafiken sind in leicht angepasster Form dieser Arbeit entnommen. Mit dem beschriebenen System wurde die Finalrunde des CeBIT Innovation Award 2015 erreicht.

8.2 Verwandte Arbeiten

Eine Integration von Gestensteuerung und freihändiger Texteingabe für mobile Anwendungen wurde in der existierenden Literatur noch nicht behandelt, stellt also ein Alleinstellungsmerkmal dieser Dissertation dar. Die Erkennung von Gesten im Kontext natürlicher Benutzerschnittstellen wird dagegen schon länger untersucht (siehe Abschnitt 1.3). Im Folgenden wird das Problem der Gestendetektion noch einmal einmal tiefgehender als in Abschnitt 1.3 im Kontext existierender Arbeiten betrachtet.

Um das Problem der Gestendetektion im kontinuierlichen Betrieb zu lösen, wurden verschiedene Verfahren für die nutzergesteuerte Segmentierung vorgeschlagen. Eine Möglichkeit stellt eine spezielle Startgeste dar, welche die eigentliche Gestenerkennung startet. Dieser Ansatz wird beispielsweise beim

kommerziell erhältlichen Myo-Eingabegerät[1] verfolgt. Eine andere Möglichkeit ist die Einführung einer Nebenbedingung unter der die Geste ausgeführt werden muss. Williamson et al. [WCB11] benutzen einen Magnetometer, um Eingaben durch Armgesten nur zuzulassen, wenn der Arm parallel zur Erdoberfläche gehalten wird. Eine nutzergesteuerte Segmentierung bedeutet immer eine Einschränkung für den Nutzer, entweder durch einen zusätzlichen Schritt, der die Interaktion umständlicher macht, oder durch eine Bedingung, welche die Interaktion weniger flexibel und natürlich macht.

Auch die Möglichkeit einer automatischen Detektion wurde in der vorhandenen Literatur untersucht. Es lassen sich zwei Ansätze zur automatischen Detektion unterscheiden, eine explizite Segmentierung durch eine Identifikation von potentiellen Gestensegmenten im kontinuierlichen Datenstrom und anschließender Erkennung [Pre08, AAS+09, RLNS10, PLH+11] oder eine implizite Segmentierung durch eine kontinuierliche Erkennung [LK99].

Das zweistufige explizite Verfahren ist bei geeigneter Wahl des Detektionsverfahrens deutlich weniger rechenaufwändig und damit energieeffizienter als eine implizite Segmentierung über eine kontinuierliche Erkennung. Allerdings können Fehler aus der ersten Stufe in der weiteren Verarbeitung nicht mehr korrigiert werden. Für mobile, am Körper tragbare Systeme wird aufgrund des hohen Rechen- und damit auch Energieaufwandes in der Regel eine explizite Segmentierung gewählt. Dabei werden in der Regel über eine schwellwertbasierte Methode direkt auf dem Sensorknoten am Arm mögliche Gestensegmente detektiert und zur weiteren Klassifikation an ein leistungsstärkeres Mobilgerät gesendet [PLH+11, Pre08, RLNS10]. Über ein Null-Klassen-Modell werden in der Erkennung falsch-positiv Segmente verworfen. Der Fokus der genannten Arbeiten liegt auf der Energieeffizienz. Die Evaluation beschränkt sich auf die getrennte Auswertung der Fehlerrate für isoliert aufgenommene Gesten und der Analyse der falsch-positiv Rate während einfacher Aktivitäten und ohne Ausführung von Gesten.

Im Rahmen dieser Arbeit wird die Evaluation der Gestendetektion und Erkennung als Gesamtsystem unter realistischen Bedingungen durchgeführt und unterschiedliche Methoden aus der Literatur verglichen. Zusätzlich wird ein System, welches eine Gestensteuerung mit freihändiger Texteingabe kombiniert, eingeführt.

[1] www.thalmic.com

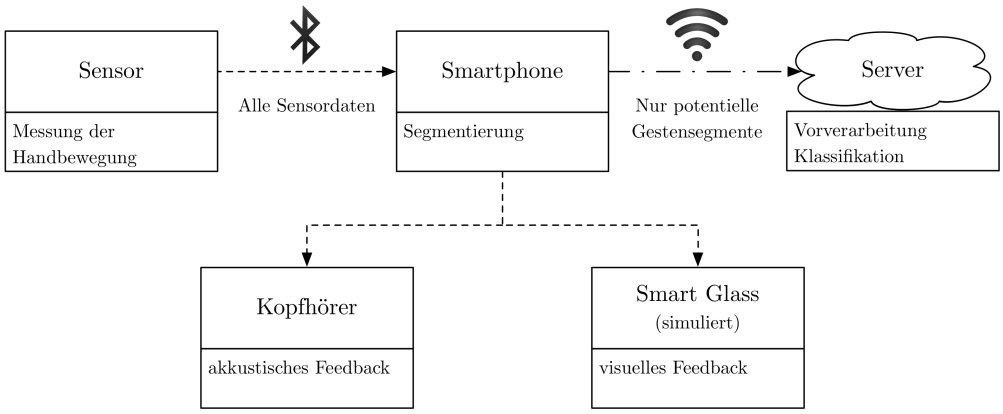


Abbildung 8.1 – Die Systemkomponenten des *Airactions*-Systems im Überblick, die obere Reihe beinhaltet die Komponenten zur Eingabeverarbeitung, die untere Reihe beinhaltet die Ausgabekomponenten.

8.3 Systemarchitektur

Das System besteht aus mehreren Komponenten, welche in Abbildung 8.1 schematisch dargestellt sind. Die Bewegungserfassung erfolgt durch das in Abschnitt 3.1 beschriebene Armband. Die Sensordaten werden via Bluetooth an ein Android Smartphone oder Tablet übertragen und dort in relevante Signalbereiche vorsegmentiert. Potentielle Gesten- oder Schriftsegmente werden an einen Erkennungsserver in der Cloud gesendet, auf dem die eigentliche Erkennung der Geste oder des geschriebenen Textes erfolgt. Das Ergebnis wird zurückgesendet, das Smartphone leitet die passende Aktion (Befehl, Eingabe) ein und stellt die Ausgabe optional im Sichtfeld des Benutzers dar. Die Augmented-Reality Ausgabe wird auf einem Bildschirm simuliert.

Im Gegensatz zu dem in Kapitel 7 vorgestellten *Airwriting*-System, für welches sich der Nutzer in Funkreichweite eines Laptops befinden muss, ist die Zielsetzung des *Airactions*-Systems, dem Nutzer größtmögliche Mobilität zu ermöglichen. Die Auslagerung der Erkennung in die Cloud ermöglicht es auf einfache Art, ein vollständig mobiles System zu implementieren. Grundsätzlich ist auch eine Portierung des in Kapitel 4 beschriebenen Erkennersystems auf ein Mobilgerät möglich. Entsprechende optimierte Algorithmen mit reduziertem Rechen- und Speicheraufwand existieren [SS09]. Allerdings geht dies in der Regel mit einem hohen Implementierungsaufwand einher und bietet keine Vorteile in der Beantwortung der im Rahmen dieser Arbeit untersuchten Fragestellungen. Eine durch die Auslagerung in die Cloud entstehende,

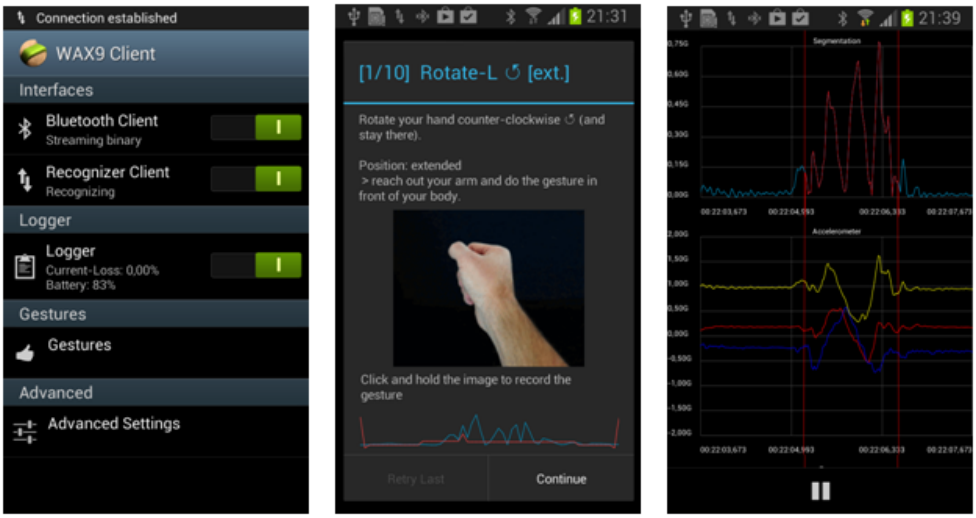


Abbildung 8.2 – Bildschirmfotos der *Airactions* Smartphone Applikation. Das linke Bild zeigt die Übersicht der Applikationseinstellungen, das mittlere Bild die Erfassung von Daten und das rechte Bild eine Echtzeitvisualisierung der erfassten Sensorsignale.

spürbare Latenz konnte nicht festgestellt werden. Auf dem Server können zudem mehrere Erkennungermodule parallel betrieben werden. Damit können neben dem Gestenerkenner auch *Airwriting*-Erkenner mit unterschiedlichen Vokabularen vorgehalten werden. Es werden beispielsweise spezialisierte Erkenner für die Kontaktliste und für die Medienbibliothek verwendet, deren Wörterbücher ausschließlich die Kontaktnamen oder entsprechend die Namen der Einträge in der Medienbibliothek enthalten.

Die zentrale Applikation läuft auf dem Smartphone. Hier werden alle Komponenten miteinander verbunden. Die Applikation modelliert den Applikationskontext und definiert darüber welche Geste im aktuellen Kontext welche Funktion auslöst. Ergibt eine Geste im aktuellen Interaktionskontext keinen Sinn, dann wird sie im Falle der Erkennung verworfen. Über ein Trainingsmodul können Trainingsdaten für neue Gesten aufgenommen und an den Server übertragen werden. Dies erlaubt es Nutzern, auf einfache Art und Weise eigenständig neue Gesten zu definieren. Abbildung 8.2 zeigt Screenshots der Applikation, die exemplarisch auf Android implementiert wurde.

8.4 Datensatz

Für die Evaluation der Verfahren wurde eine Gestenmenge definiert und ein Datensatz unter realistischen Bedingungen aufgenommen. Im Folgenden werden die verwendete Gestenmenge und die Kriterien für ihre Zusammenstellung sowie der erstellte Datensatz eingeführt.

8.4.1 Gestenkatalog

Für die Interaktion mit Handgesten existiert derzeit noch kein etablierter Katalog an Gesten und deren Abbildung auf Funktionalität. Die Menge möglicher Gesten wurde auf einhändige Gesten eingeschränkt. Dies ermöglicht eine einfache Erfassung mit einem Sensorarmband an der dominanten Hand, die auch für die Texteingabe mittels *Airwriting* benutzt wird. Da die Inertialsensorik keine Erfassung der relativen Position der Sensoren untereinander zulässt, sind zudem Gesten, die durch die relative Position und Bewegung der Hände zueinander definiert sind, schwierig zu erfassen und damit auch schwierig zu erkennen.

Auf der Basis vorhandener Literatur wurden insgesamt 15 verschiedene Kommandogesten definiert, die sich a) an der bereits bekannten Bedienung von Touchscreen-Oberflächen und b) an Metaphern der physikalischen Welt orientieren. So kann beispielsweise mittels Wischgesten durch Menüs navigiert werden und ein Griff an das Ohr kann die Annahme oder die Initiierung eines Gesprächs in Anlehnung an die reale Benutzung eines Telefons signalisieren. Abbildung 8.3 zeigt den verwendeten Gestenkatalog.

Sehr häufig werden in der Literatur Rotationen des Handgelenks verwendet [JALT08, PLH+11, RLNS10, WCB11]. In dieser Arbeit werden die vier Rotationsgesten *Rotate-R*, *Rotate-L*, *Rotate-RL*, *Rotate-LR* benutzt, welche jeweils als schnelle Drehung des Handgelenks um ca. 90° nach links (L) oder rechts (R) und gegebenfalls wieder zurück (LR, RL) definiert sind.

Auch eine kreisförmige Bewegung in beide Richtungen, *Circle-R* und *Cirle-L*, ist eine häufig genutzte Geste [LK99, LZWV09, RLNS10]. Zusätzlich werden noch die Gesten *Wirl-R* und *Wirl-L* definiert, welche eine kreisförmige Bewegung aus dem Handgelenk, also ohne den Unterarm zu bewegen, beschreiben.

Als Entsprechung der Touchscreen-Wischgesten werden die Gesten *Swipe-R* und *Swipe-L* definiert. Sie beschreiben eine Wischbewegung nach links oder rechts aus dem Handgelenk heraus. Auch ein einfacher Klick oder Doppel-

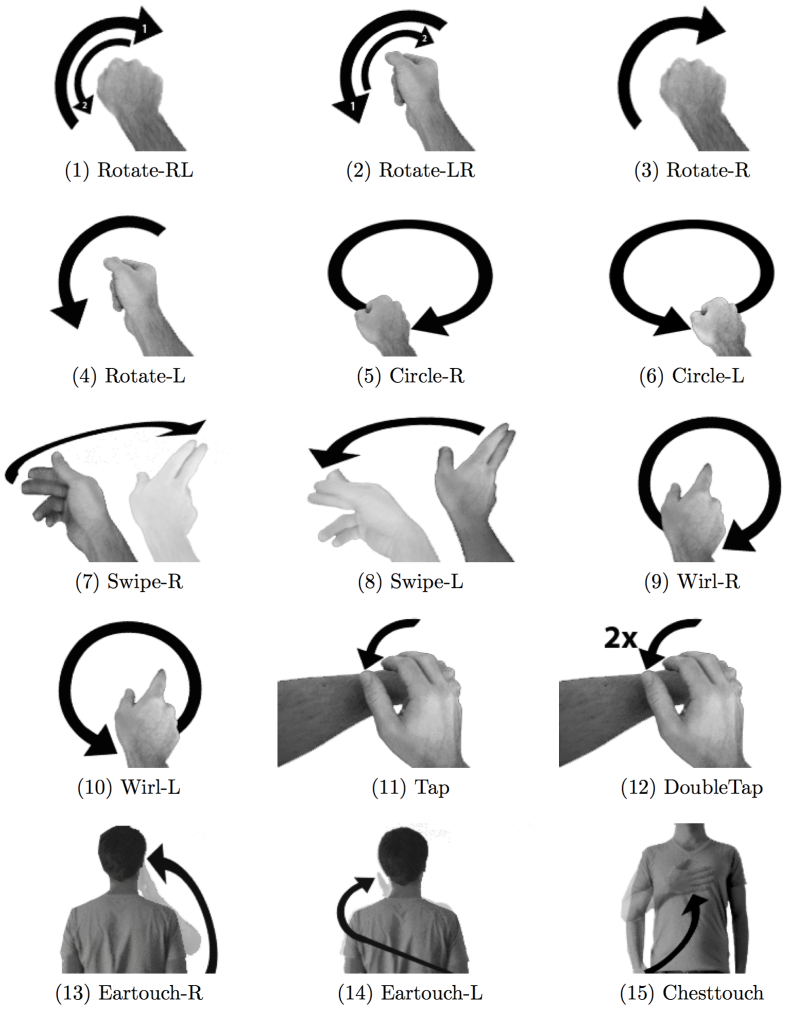


Abbildung 8.3 – Übersicht der verwendeten Gesten im Gestenkatalog.

klick wurde als Geste realisiert. Die Gesten *Tap* und *DoubleTap* sind als ein- oder zweimaliges Klopfen mit dem Zeigefinger gegen eine beliebige Unterlage definiert (*Tap* ebenfalls benutzt in [RB10]).

Die Benutzung eines Telefons wird durch die Geste *Eartouch-R* dargestellt, bei der sich der Nutzer mit der rechten Hand an sein rechte Ohr greift [JALT08, PLH+11, RLNS10]. Zusätzlich wird auch der Griff mit der rechten Hand an das linke Ohr mit der Geste *Eartouch-L* eingeführt. Die Bewegung der Hand zur Brust wird als Geste *Chesttouch* aufgenommen [RLNS10].

Datensatz	Inhalt	Probanden	Größe
Gesten			
Gesten.E	Einzelne Gesten	10	3000 Gesten
Gesten.A	Gesten und Aktivitäten	10	2h 7m 28s

Tabelle 8.1 – Überblick über den erstellten Gestendatensatz (Gesten.E und Gesten.A).

8.4.2 Datenaufnahme

Für die Evaluation wurden zwei unterschiedliche Datensätze aufgenommen. Der Datensatz Gesten.E enthält Aufnahmen einzelner Gesten und dient zum Training und Test der Gestenerkennung. Der Datensatz Gesten.A enthält kontinuierliche Aufnahmen von Alltagsaktivitäten mit sporadisch ausgeführten Gesten. Dieser Datensatz dient zur Evaluation des Gesamtsystems aus Erkennung und Detektion. An der Datenaufnahme nahmen zehn Probanden im Alter von 21 bis 34 Jahren teil (2w/8m). Alle Probanden waren Rechtshänder. Für Linkshänder müsste ein separates System erstellt werden, da viele Bewegungen gespiegelt sind und sich damit beispielsweise die Rotationsrichtung der Handrotationsgesten ändert. Es ist aber zu erwarten, dass ein solches System bei gleicher Trainingsdatenmenge eine vergleichbare Leistung erreicht, wie ein System für Rechtshänder. Jeder Proband absolvierte zuerst die Aufnahmen der einzelnen Gesten und anschließend die Aufnahmen der Gesten, eingebettet in Alltagsaktivitäten. Jede Person führte zwei Aufnahmesitzungen durch, zwischen denen ein Zeitraum von mindestens zwei Wochen lag.

Die Software für die Datenaufnahme läuft unter Android auf einem Smartphone, für die Datenaufzeichnung ist daher kein stationärer Rechner nötig. Die Probanden wurden kurz in die Software eingewiesen und die auszuführenden Gesten wurden erklärt. Danach führten die Probanden die Aufnahmen selbstständig durch.

Für die Aufnahmen der Einzelgesten zeigt die Software die auszuführende Geste durch eine textuelle Beschreibung und ein Video an. Der Nutzer berührt den Bildschirm auf der Fläche der Videodarstellung, um die eigene Gestenausführung zu segmentieren. Das Gerät wird dabei in der nicht dominanten Hand gehalten, mit der man auch die Applikation bedient. Die Geste wird mit der dominanten Hand ausgeführt. Solange der Nutzer mit seinem Finger die Fläche der Videodarstellung berührt, wird aufgenommen. Für jede Geste werden zwei Ausführungsvarianten abgefragt, mit hängenden Arm und

im Ellbogengelenk angewinkeltem Arm. In der hängenden Variante wird die Geste mit nach unten hängendem Arm ausgeführt. Bei Gesten, die durch eine Bewegung zu einem Zielpunkt definiert sind, wie beispielsweise die *Eartouch* Gesten, wird die Bewegung aus der nach unten hängenden Armposition gestartet. In der im Ellbogengelenk angewinkelten Variante ist der Unterarm parallel zur Erdoberfläche und der Oberarm am Körper anliegend.

Für die Aufnahmen der Aktivitäten können die Probanden beliebige Gesten zu beliebigen Zeitpunkten ausführen. Die Gesten werden wie für die Einzelaufnahmen durch das Berühren des Bildschirms segmentiert. Nach der Segmentierung haben die Probanden die Möglichkeit, die ausgeführte Geste durch Auswahl aus einer Liste zu annotieren. Die Probanden wurden angehalten, verschiedene Aktivitäten auszuführen und dabei sporadisch Gesten einzustreuen. Die ausgeführten Aktivitäten wurden nicht beschränkt, sie umfassten beispielsweise Gehen, Laufen, Sitzen und manuelle Tätigkeiten wie Trinken, Rechnerarbeit oder Türenöffnen. Das mobile Aufnahmesystem und die einfache Annotation durch den Probanden selbst gestatteten hier eine relativ große Freiheit in der Wahl des Aufnahmeortes und der Aktivitäten. Einschränkend wirkt sich allerdings die Verwendung der nicht dominanten Hand zur Segmentierung und Annotation aus. Aktivitäten, für welche die nicht dominante Hand notwendig ist, wie beispielsweise Fahrradfahren, müssen für die Segmentierung unterbrochen werden. Diese Restriktionen könnten in weiteren Arbeiten beispielsweise durch eine Annotation mit gesprochener Sprache, in Verbindung mit automatischer Spracherkennung, überwunden werden. Die durchschnittliche Aufnahmedauer pro Proband betrug 13 min, in denen jeder Proband durchschnittlich 26 Gesten ausführte.

In Abbildung 8.4 ist die Häufigkeitsverteilung der Gesten in dem Datensatz zu sehen. Insgesamt beinhaltet der Datensatz 261 Gesten. Die Verteilung ist sehr unbalanciert, da die Nutzer nach eigenem Ermessen auswählen konnten, welche Gesten sie ausführen. Ein balancierter Datensatz wäre nur möglich, indem die Nutzer durch das System aufgefordert werden, eine bestimmte Geste auszuführen. Dies führt aber automatisch zu einer Unterbrechung der aktuellen Nutzeraktivität kurz bevor die Geste ausgeführt wird und folglich sind die Gesten nicht mehr natürlich in Alltagsaktivitäten und Bewegungen eingebettet. Aus diesem Grund wurde auf die Erstellung eines balancierten Datensatzes verzichtet. Die Streuung im Datensatz lässt sich möglicherweise durch Präferenzen für bestimmte Gesten erklären. Dies wäre ein interessanter weiterer Untersuchungsgegenstand. Der im Rahmen dieser Arbeit aufgenommene Datensatz lässt aufgrund des Studienaufbaus allerdings keine validen Schlüsse hinsichtlich der Nutzerpräferenz zu.

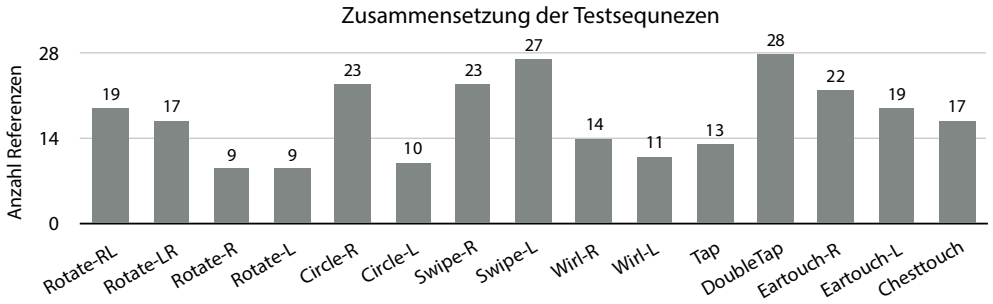


Abbildung 8.4 – Anzahl der aufgenommenen Gesten, die in Alltagsaktivitäten eingebettet sind (Datensatz Gesten.A), über alle zehn Probanden akkumuliert.

8.5 Detektion

Die Detektion der Kommandogesten unterscheidet sich von der Detektion der Handschrift in zwei wesentlichen Punkten. Erstens sind einzelne Gesten in der Regel deutlich kürzer, sie entsprechen in ihrer Dauer ungefähr der Dauer eines kurzen Buchstabens. Zweitens lassen sich, bedingt durch die sehr unterschiedlichen Bewegungen der Gesten und die kurze Dauer, im Gegensatz zur Handschrift keine charakteristischen Frequenzmerkmale bestimmen. Eine direkte Übertragung des Verfahrens zur Detektion von *Airwriting* (siehe Abschnitt 7.2) ist entsprechend nicht möglich. Im Folgenden wird ausgehend von bestehender Literatur ein Verfahren zur Detektion von Gesten eingeführt. Zuerst werden geeignete Merkmale für die Gestendetektion beschrieben und anschließend ein schwellwertbasiertes Verfahren für die Detektion und Segmentierung der Gesten.

8.5.1 Merkmalsextraktion Detektion

Aus dem sechsdimensionalen Beschleunigungs- und Drehratensignal werden zwei etablierte Merkmale für die Detektion verglichen. Eines basiert auf der absoluten Beschleunigung und eines auf deren Änderungsrate.

Die sogenannte *instantanious Hand Force* (HF) bezeichnet die gravitationsbereinigte euklidische Norm der Beschleunigung und wird in [PLH+11, RLNS10] als Merkmal für die Detektion verwendet. Für einen Beschleunigungsvektor (x_t, y_t, z_t) zum Zeitpunkt t berechnet sich diese durch

$$\mathrm{HF} = |\sqrt{x_t^2 + y_t^2 + z_t^2} - 1\,\mathrm{g}| \ .$$

Bewegt sich der Sensor nicht, dann gilt HF = 0. Durch den Abzug der Gravitation *nach* Berechnung der Norm ergibt sich für eine parallel zur Gravitation auftretende Beschleunigung ein unterschiedlicher Wert für HF als für eine orthogonal zur Gravitation auftretende Beschleunigung. Dies macht die Wahl geeigneter Schwellwerte schwierig.

In [Pre08] wird ein auf der Änderungsrate basierendes Merkmal H verwendet. Es berechnet sich durch die euklidische Norm der Differenzen aufeinanderfolgender Sensorwerte zum Zeitpunkt $t-1$ und t als Approximation der Ableitung gemäß

$$d_t = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2}.$$

Über einen exponentiellen gleitenden Mittelwert wird das Resultat geglättet und H berechnet sich entsprechend durch

$$H_t = \alpha \cdot d_t + (1 - \alpha) \cdot H_{t-1}$$

mit $\alpha = 0,5$. Das Merkmal H ist im Gegensatz zu HF unabhängig von der Orientierung des Sensors. Aufgrund der Auswahl der Gesten, welche sowohl Bewegung parallel als auch orthogonal zur Gravitation enthalten, kann davon ausgegangen werden, dass ein orientierungsunabhängiges Merkmal zu besseren Resultaten führt. Dies wird in Abschnitt 8.7.2 auch experimentell bestätigt.

8.5.2 Segmentierung

Die Segmentierung erfolgt über eine doppelte Schwellwertbildung durch eine *ForwardBackward Movement Detection* [RLNS10]. Ein relativ hoher Schwellwert T_S wird verwendet, um Bewegungen zu detektieren und zwei relativ niedrige Schwellwerte T_R und T_V werden genutzt, um das Bewegungssegment präzise einzugrenzen. Das Verfahren wird in Abbildung 8.5 illustriert.

Wird der Schwellwert T_S zum Zeitpunkt t_0 überschritten wird ein neues potentielles Gestensegment erstellt. Nun wird das Signal von t_0 aus nach vorne (Vorwärts) und hinten (Rückwärts) durchlaufen, um die exakten Grenzen des Segments zu bestimmen. Wenn der jeweilige Schwellwert T_R und T_V für mindestens T_L Werte unterschritten wurde, wird die entsprechende Segmentgrenze festgesetzt. Über den Parameter T_L wird definiert, wie lange der Schwellwert unterschritten werden kann, ohne dass eine Segmentgrenze gesetzt wird. Dies verhindert eine Stückelung des Segments in mehrere Segmente, falls der Schwellwert nur für kurze Zeit unterschritten wird. Für die

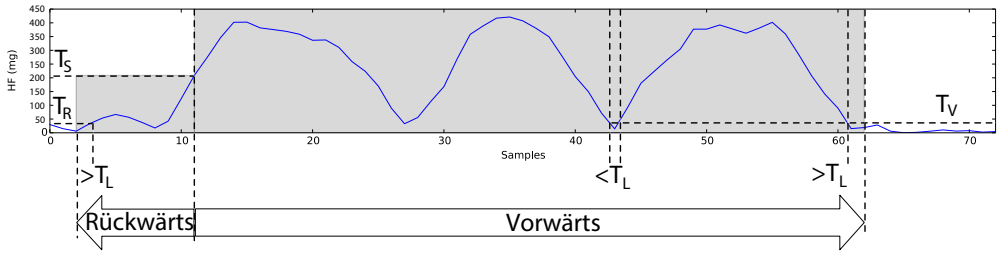


Abbildung 8.5 – Illustration des Segmentierungsalgorithmus: Bei Überschreiten des Schwellwerts T_S wird der Anfang des Segments über den Schwellwert T_R rückwärts und das Ende des Segments über den Schwellwert T_V vorwärts bestimmt. Die maximal erlaubte Länge der Schwellwertunterschreitung ist durch T_L gegeben.

Merkmal	T_S	T_R	T_V	T_L
H	195	41	53	0
HF	570	40	30	40

Tabelle 8.2 – Werte der Segmentierungsparameter.

Parameter wurden die in Tabelle 8.2 angegebenen optimalen Werte empirisch ermittelt.

8.6 Klassifikation

8.6.1 Modellierung

Jede Geste wird mit einem HMM, gemäß des in Kapitel 4 beschriebenen Basissystems, modelliert. Es wurden HMMs mit neun Zuständen und sechs Mixturkomponenten gewählt. Zusätzlich zu den Gesten wird ein Nullklassen-Modell eingeführt. In der Spracherkennung wird ein solches Modell häufig als *garbage*-Modell bezeichnet. Es wird auf allen Gestendaten trainiert, um ein möglichst unspezifisches aber auf die Bewegungsdaten passendes Modell zu erhalten. Wird ein Bewegungssegment detektiert, welches keine der definierten Gesten enthält, sollte im Idealfall das Null-Klassen-Modell die höchste Wahrscheinlichkeit liefern und damit die falsch-positiv Detektion korrigieren. Da die Null-Klasse keinen charakteristischen zeitlichen Verlauf aufweist, wird für die Modellierung ein HMM mit einem Zustand und 12 Mixtur-

komponenten verwendet. Der Wert wurde experimentell ermittelt, allerdings unterscheiden sich die Ergebnisse für einen Bereich von 4 bis 15 Mixturkomponenten nicht signifikant.

8.6.2 Merkmalsextraktion Klassifikation

Für den i-ten Abtastwert bezeichnet $\boldsymbol{x}_i$ das sechsdimensionale Sensorsignal. Analog zur Detektion werden die Signale über einen exponentiellen gleitenden Mittelwert geglättet. Das geglättete Signal wird mit $\tilde{\boldsymbol{x}}$ bezeichnet und berechnet sich gemäß

$$\tilde{\boldsymbol{x}}_i = \alpha \cdot \boldsymbol{x}_i + (1 - \alpha) \cdot \tilde{\boldsymbol{x}}_{i-1}$$

mit $\alpha = 0.5$. Im Gegensatz zu der Schrifterkennung geht es bei den eingeführten Steuergesten meist nicht nur darum, eine bestimmte Bewegungstrajektorie auszuführen, sondern auch darum, wie schnell oder ruckartig die Bewegung ausgeführt wird. Aus diesem Grund werden zwei weitere Merkmale berechnet, die Approximationen der Ableitung und des Integrals des Sensorsignals darstellen. Die Ableitung wird über die Differenz aufeinanderfolgender Sensorwerte über

$$\Delta\tilde{\boldsymbol{x}}_i = \tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_{i-1}$$

angenähert, das Integral über die Summation der letzten k Werte mit

$$Sum_{\tilde{\boldsymbol{x}}_i} = \sum_{j=i-k+1}^{i} \tilde{\boldsymbol{x}}_j \ .$$

Für k wurde empirisch ein optimaler Wert von $k = 15$ ermittelt. Alle Merkmale wurden nach der Berechnung z-normalisiert und in einen 18-elementigen Merkmalsvektor vereint.

8.7 Evaluation

Die Evaluation gliedert sich in zwei Teile. Zuerst wird die Akkuratheit auf den Einzelgesten bestimmt und anschließend das Gesamtsystem, bestehend aus den zwei Stufen Detektion und Gestenerkennung, evaluiert.

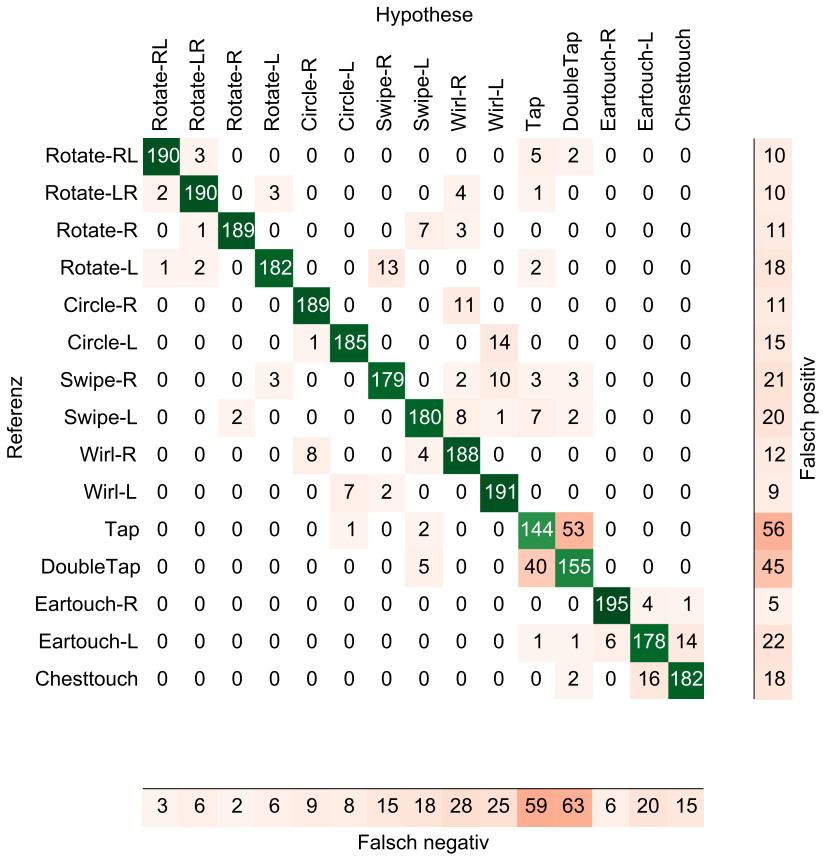


Abbildung 8.6 – Konfusionsmatrix der Einzelgestenerkennung auf dem Gesten.E-Datensatz. Grüne Felder markieren korrekte Klassifikationen, rote Felder markieren Fehler.

8.7.1 Einzelgesten

Die Fehlerrate auf den Einzelgesten wurde für den personenunabhängigen Fall evaluiert. Hierfür wurde eine Kreuzvalidierung über die Probanden durchgeführt. Es wird eine Fehlerrate von 9,4 % erreicht. In Abbildung 8.6 ist die Konfusionsmatrix dargestellt. Die *Tap*- und *DoubleTap*-Gesten werden mit Abstand am häufigsten verwechselt. In den folgenden Experimenten (Abschnitt 8.7.2) zum Gesamtsystem wird sich zusätzlich zeigen, dass diese Gesten in Kombination mit der Detektion zu weiteren Fehlern führen. Dies sollte bei dem Design zukünftiger Schnittstellen in Betracht gezogen werden, ein Verzicht auf diese Gesten erscheint naheliegend. Um den Effekt eines Verzichts zu untersuchen, wird das Gesamtsystem in Abschnitt 8.7.2 zusätzlich mit einer um die *Tap* und *DoubleTap* Gesten reduzierten Gestenmenge eva-

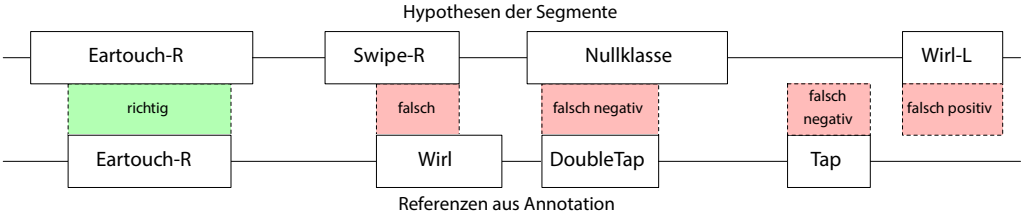


Abbildung 8.7 – Mögliche Fehlervarianten für das Gesamtsystem.

luiert. Weitere Verwechslungen treten, wie zu erwarten, für sich ähnelnde Gesten auf. So wird beispielsweise *Chesttouch* häufig mit *Eartouch-L* verwechselt und die *Wirl* Gesten häufig mit den *Circle* Gesten. Da die *Swipe* Geste mit einer Verschiebung der Elle gegen die Speiche und damit einer Rotation des Handgelenks einhergeht, kommt es zu den auf den ersten Blick eher überraschenden Verwechslungen zwischen den *Swipe* und *Rotate* Gesten. Die Resultate dieser Evaluation können, in Verbindung mit der in Kapitel 8.7.2 vorgestellten Evaluation des Gesamtsystems, als Grundlage für die Auswahl und die Zuordnung von Gesten zu Funktionen in zukünftigen Schnittstellen dienen.

Wird die Null-Klasse in die Evaluation miteinbezogen, steigt die Fehlerrate auf 11,5 % an, da in einigen Fällen Gesten fälschlicherweise als Null-Klasse klassifiziert werden. In den nachfolgenden Analysen des Gesamtsystems, bestehend aus Detektion und Klassifikation, wird allerdings die Notwendigkeit der Verwendung des Nullklassenmodells deutlich.

8.7.2 Gesamtsystem

Für die Evaluation des Gesamtsystems werden Detektion und Erkennung gemeinsam ausgeführt. Die Auswertung erfolgt wie für die Einzelgesten für den personenunabhängigen Fall über eine Kreuzvalidierung. Die Gestenmodelle werden wie für die Einzelgesten auf den Daten des Datensatzes Gesten.E trainiert. Der Test erfolgt ausschließlich auf den Daten des Datensatzes Gesten.A, der sporadische Gesten eingebettet in Alltagsaktivitäten enthält. Als Metriken werden Trefferquote, Genauigkeit und F-Maß berechnet. Eine korrekte Erkennung ist gegeben, wenn ein detektiertes Segment mit einem Referenzsegment überlappt und die Geste des Referenzsegments erkannt wird. Es werden drei Typen von Fehlern, die in Abbildung 8.7 grafisch dargestellt sind, unterschieden:

Merkmal	Trefferquote	Genauigkeit	F-Maß
H	81,2	70,9	75,7
HF	66,3	80,8	72,8

Tabelle 8.3 – Ergebnisse der Detektion und Erkennung auf dem Gesten.A-Datensatz für verschiedene Merkmale in der Detektion.

1. Gestenfehler: Ein detektiertes Segment überlappt mit einem Referenzsegment, aber die falsche Geste wurde erkannt. (falsch)
2. Falscher Alarm: Ein Segment wurde detektiert und eine Geste wurde erkannt, aber es gibt kein überlappendes Segment in der Referenz. (falsch-positiv)
3. Verpasser: Es wurde kein Segment detektiert oder es wurde ein Segment detektiert und die Null-Klasse erkannt, obwohl es ein Gestensegment in der Referenz gibt. (falsch-negativ)

Damit ändert sich die Berechnung der Trefferquote und der Genauigkeit zu der in Abschnitt 2.6.1 angegebenen Definition. Die Gestenverwechslungen (Fehlertyp falsch) werden sowohl für die Berechnung der Genauigkeit als auch der Trefferquote berücksichtigt und jeweils zu der Anzahl der falsch-positiven und falsch-negativen Ergebnissen hinzugerechnet. Abbildung 8.8 stellt die Berechnung der Metriken grafisch dar.

Tabelle 8.3 gibt die Ergebnisse für die Merkmale HF und H, die in der Detektion verwendet werden, an. Mit dem Merkmal H wird insgesamt ein besseres Resultat erreicht. Dies bestätigt die Hypothese, dass für die gewählte Menge an Gesten ein Merkmal, das nicht von der Sensororientierung abhängt, besser geeignet ist. Für die weiteren Betrachtungen wird daher ausschließlich die Detektion anhand dieses Merkmals verwendet.

In Abbildung 8.9 ist die Konfusionsmatrix für die Evaluation des Gesamtsystems dargestellt. Es ist deutlich zu sehen, dass Verwechslungen zwischen einzelnen Gesten relativ selten auftreten. Die häufigsten Verwechslungen gibt es zwischen *Tap* und *DoubleTap* (4 Gestenfehler) sowie *Eartouch-L* und *Chesttouch* (3 Gestenfehler). Den weitaus größten Teil der Fehler machen allerdings die falsch-positiven (falscher Alarm) und falsch-negativen (Verpasser) Fehler aus. Ein falscher Alarm tritt mit Abstand am häufigsten für die *Tap*, *DoubleTap* und die *Wirl-L* Geste auf. Verpasser treten gleichmäßiger verteilt über die Gesten auf.

In 857 Fällen wurde eine Geste fälschlicherweise detektiert, dann allerdings als Null-Klasse erkannt. Der Fehler in der Detektion wurde also durch das

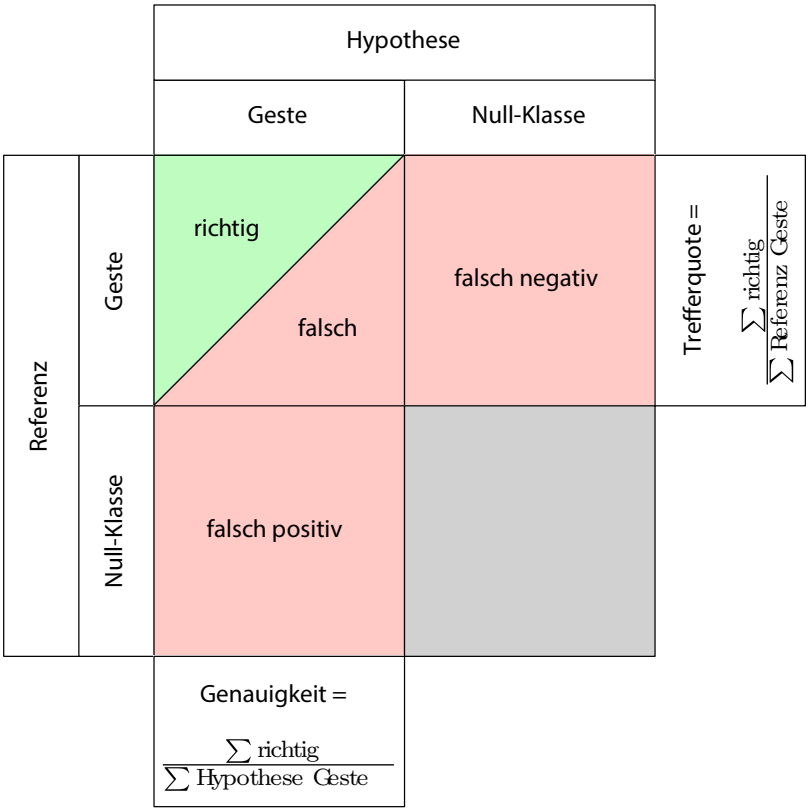


Abbildung 8.8 – Berechnung der Genauigkeit und Trefferquote für das Gesamtsystem zur Gestendetektion und Erkennung.

Null-Klassen Modell korrigiert. In 73 Fällen wurde dagegen für eine fälschlicherweise detektierte Geste auch eine Geste statt der Null-Klasse erkannt. Das bedeutet, in 92 % der Fälle, in denen fälschlicherweise eine Geste detektiert wurde, konnte der Fehler über das Modell der Null-Klasse korrigiert werden. Dies zeigt deutlich die Notwendigkeit, ein Null-Klassen-Modell zu benutzen. Allerdings existiert Optimierungspotential, da eine fälschliche Erkennung der Null-Klasse zu insgesamt 35 Verpassern führte, das entspricht 13,4 % aller ausgeführten Gesten.

Die Ergebnisse der Auswertung sind im Kontext möglicher Anwendungen zu interpretieren. In der Regel hat ein falsch-negativ Fehler, also ein Verpasser, die geringste negative Auswirkung auf die Interaktion. Der Benutzer kann die Geste theoretisch beliebig oft wiederholen, bis sie korrekt erkannt wird. Die Effektivität sinkt nicht, außer das System erkennt die Geste niemals korrekt. Die Folge einer zu geringen Trefferquote ist allerdings im Regelfall eine geringe Nutzerzufriedenheit und eine geringe Effizienz. Falsch-positiv Fehler

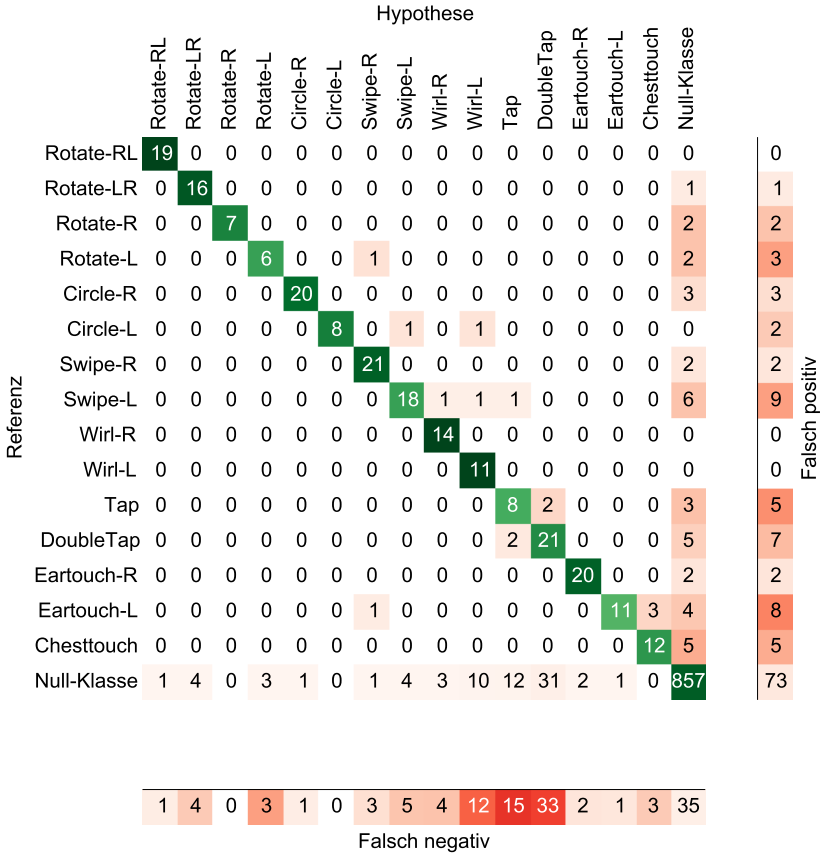


Abbildung 8.9 – Konfusionsmatrix für das Gesamtsystem. Die grünen Felder markieren korrekte Erkennungen, die roten Felder markieren Fehler.

sind in der Regel deutlich kritischer, da sie unter Umständen eine Aktion ausführen, obwohl vom Benuzter zu diesem Zeitpunkt überhaupt keine Aktion intendiert war. Ebenfalls kritisch sind Verwechslungen zwischen Gesten, hier wird unter Umständen eine Aktion ausgeführt, die nicht der vom Nutzer intendierten Aktion entspricht und somit eine Korrektur erforderlich macht. Das Systemverhalten ist damit für den Nutzer nicht mehr vorhersagbar.

Eine Analyse der Ergebnisse und weitere informelle Tests ergaben, dass die *Tap* und *DoubleTap* Gesten häufig bereits beim Ablegen der Hand auf den Tisch sowie dem Berühren oder der Manipulation eines Objekts auftreten. Die Gesten eignen sich also nur bedingt für den Einsatz. Werden die Gesten aus dem Gestenkatalog entfernt, erhöht sich die Genauigkeit von 70,9 % auf 81 %, die Trefferquote von 81,2 % auf 83,2 % und das F-Maß damit von 75 % auf 82,4 %.

Die Fehlerraten sind in Bezug auf eine praktische Anwendung noch relativ hoch, allerdings legt die Evaluation ein Szenario zugrunde, in dem alle Gesten zu jedem beliebigen Zeitpunkt erkannt werden können, also in einer Anwendung mit einer Funktion belegt sind. In den meisten Anwendungsszenarien wird aber nicht jede Geste zu jedem Zeitpunkt mit einer ausführbaren Aktion verknüpft sein. Beispielsweise könnten die *Swipe* Gesten mit der Navigation durch Menüeinträge verknüpft und entsprechend nur im Kontext eines dargestellten Menüs aktiviert sein. Aktionen, die zu jedem beliebigen Zeitpunkt durchgeführt werden können, sollten entsprechend durch Gesten mit geringer Anzahl an falschen Alarmen abgebildet werden. Befindet sich der Nutzer dann in einem Interaktionskontext, in dem sein Hauptfokus auf der Interaktion mit dem System liegt, kann insgesamt von einer höheren Genauigkeit als in den dargestellten Ergebnissen ausgegangen werden. Fehler, die durch Verwechslungen zwischen den Gesten entstehen, können schließlich durch eine geeignete Wahl der Zuordnung von Geste zu Funktion gewählt werden. Leicht verwechselbare Gesten sollten immer auf Funktionen abgebildet werden, die in einem Zustand der Interaktion nicht gleichzeitig erlaubt sind. Wird eine Zuordnung von Geste zu Funktion allerdings alleinig nach diesen Kriterien vorgenommen, senkt dies die Intuitivität. Dies führt zu einem erhöhten Lernaufwand und möglicherweise zu einer geringeren Benutzerzufriedenheit.

In einem realen System besteht eine Koadaptation zwischen Nutzer und System. Der Nutzer passt sich automatisch an das interne Modell der Bewegung an, indem er die Bewegungen so ausführt, dass die Fehlerrate sinkt. Dies führt also zu einer Verringerung der Fehlerrate, verlangt vom Nutzer allerdings einen, wenn auch impliziten, Lernprozess. Dieser Faktor wurde in der vorliegenden Arbeit ausgeschlossen, um zu untersuchen wie hoch die Fehlerraten der Gestendetektion und Erkennung ohne jede Vorerfahrung des Nutzers sind. Die Koadaptation, sowohl aus Sicht des Nutzers als auch des Systems, könnten Gegenstand weiterer Untersuchungen sein.

8.8 Demonstrator

Aufbauend auf den Ergebnissen und der Architektur des *Airactions*-Systems wurde ein Demonstrator für ein System, welches Steuergesten und Airwriting integriert, entwickelt. Die Detektion und Interaktionsmodellierung erfolgt dabei auf einem Smartphone, die Erkennung von Gesten und Airwriting wird serverseitig über eine WLan Verbindung ausgeführt. Die Ausgabe wird durch

ein simuliertes Brillendisplay auf einem Monitor realisiert. Es wurden drei unterschiedliche Szenarien gewählt und implementiert:

1. Bedienung eines Musikspielers: Für den Musikspieler auf einem Smartphone wurden die elementaren Funktionen wie „Abspielen“, „Pause“, „Titel weiter“, „Titel zurück“, „Lauter“ und „Leiser“ über Kommandogesten realisiert. Da die Rückmeldung an den Nutzer für diese Aktionen akustisch erfolgt, können diese Funktionen auch ohne eine entsprechende Brille genutzt werden. Über die Eingabe von Text können auch komplexere Aufgaben, wie beispielsweise die Suche nach Interpreten und Titeln in der Musikdatenbank, ausgeführt werden. Hierbei wird zur Darstellung der Suchergebnisse auf das simulierte Brillendisplay zurückgegriffen.

2. Bedienung einer Telefonfunktion: Über die Texteingabe können Kontakte in der Kontaktliste des Smartphones gesucht und anschließend mittels Steuergesten durch das Suchergebnis navigiert und der gewünschte Kontakt ausgewählt werden (siehe Abbildung 8.10, oberes Bild). Die Annahme oder Initiierung von Anrufen kann über eine Berührung des Ohres (Eartouch), als Methapher für das Telefonieren, erfolgen.

3. Abarbeiten einer Checkliste: Eine mögliche industrielle Anwendung wird mit dem Abarbeiten einer Checkliste, beispielsweise für Inspektionen in der Qualitätskontrolle, gezeigt. Über das Brillendisplay werden Technikern nacheinander auszuführende Prozessschritte angezeigt. Per Gestenkommandos kann zum nächsten Schritt weitergeschaltet werden oder über Schrift ein Kommentar eingefügt werden, beispielsweise dass ein bestimmtes Bauteil getauscht, repariert, nachjustiert, usw. werden muss (siehe Abbildung 8.10, mittleres und unteres Bild). Typischerweise werden im industriellen Umfeld Handschuhe getragen und es gibt zahlreiche laute Umgebungsgeräusche. In solchen Szenarien ist der Einsatz von Sprachtechnologie und Touchscreens schwierig, während eine gestenbasierte Lösung von diesen Faktoren nicht beeinträchtigt wird.

In dem Demonstrator-System wird damit erstmals eine mobile, gestenbasierte Schnittstelle mit Freihandtexteingabe in den genannten drei implementierten Szenarien gezeigt.

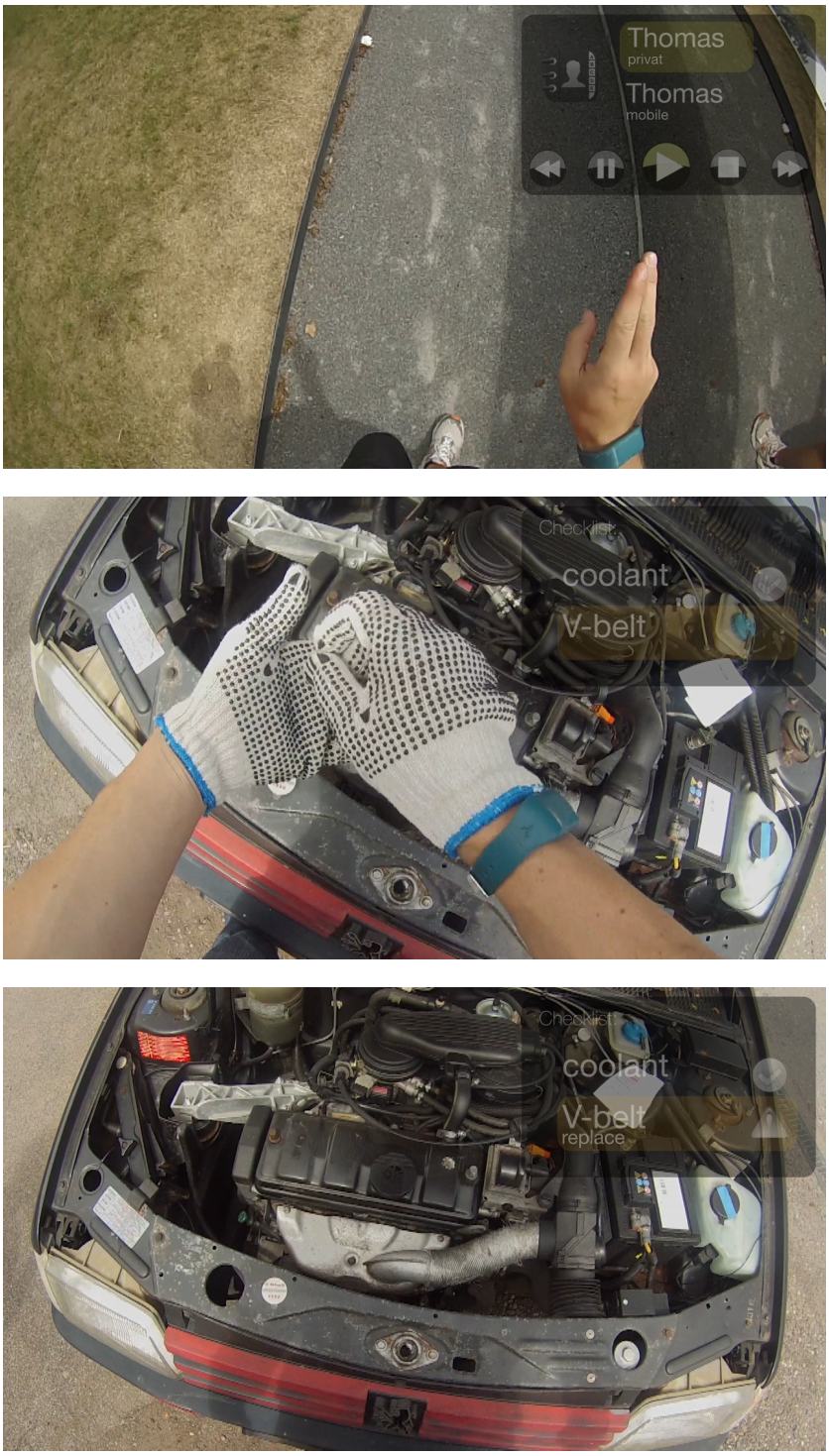


Abbildung 8.10 – Illustrationen des Airactions Demonstrators. Das obere Bild zeigt die Ausführung einer Wischgeste, um durch die Nummern eines Telefonkontakts zu navigieren. Die zwei unteren Bilder zeigen die Verwendung der Texteingabe mittels Airwriting (mittleres Bild), um in einer Aufgabenliste Annotierungen einzufügen (unteres Bild).

8.9 Zusammenfassung

In diesem Kapitel wird das Airactions-System vorgestellt, welches eine Gestensteuerung über einzelne Kommandogesten mit der Texteingabe via Airwriting verbindet. Dies ermöglicht einen weitaus höheren Funktionsumfang als bisher in der Literatur beschriebene Gestenschnittstellen aufweisen, die ausschließlich auf einer Gestensteuerung mit Einzelgesten basieren. Das System selbst läuft auf einem Smartphone und ist damit mobil einsetzbar, die rechenaufwändige Erkennung wird serverseitig ausgeführt.

Die Komponenten zur Gestensteuerung werden eingeführt und evaluiert. Die verwendete Gestenmenge wird mitsamt der Kriterien für die Auswahl beschrieben. Aufbauend auf den in der Arbeit verwendeten Verfahren zu Schrifterkennung werden die Verfahren zur Gestendetektion und Erkennung eingeführt. Die Verfahren werden an die geänderten Anforderungen angepasst und unter realistischen Bedingungen evaluiert. Die Ergebnisse werden mit Fokus auf die praktische Relevanz diskutiert und können damit als Grundlage für das Design zukünftiger gestenbasierter Schnittstellen dienen.

Abschließend wird der implementierte Airactions-Demonstrator beschrieben. Mit diesem System wurde die Finalrunde des CeBIT Innovation Awards 2015 erreicht.

Kapitel 9

Zusammenfassung und Ausblick

In diesem letzten Kapitel werden die Ergebnisse der Arbeit zusammengefasst. Die wesentlichen Beiträge werden insbesondere im Hinblick auf ihre Bedeutung für die weitere Forschung in dem Bereich des Wearable Computings *diskutiert. Abschließend werden sich ergebende Forschungsfragen für die Zukunft sowohl in technischer Hinsicht als auch im Kontext der Mensch-Maschine-Interaktion aufgeworfen und kurz umrissen.*

In der Einleitung dieser Arbeit wurde die Texteingabe als eine zentrale Herausforderung in dem Bereich des *Wearable Computings* identifiziert. Mit dieser Dissertation wird eine mögliche Lösung für dieses Problem präsentiert. In Abgrenzung zu früheren Arbeiten werden erstmalig die, in Abschnitt 1.4 benannten, wesentlichen Kriterien der Freihändigkeit (*Airwriting*), der leichten Erlernbarkeit (Handschrift), der unmittelbaren und ständigen Verfügbarkeit (Detektion) und der Effizienz (kontinuierliche Schrifteingabe) zusammen erfüllt. Damit steht neben den technischen Beiträgen und Resultaten auch die Konzeption eines innovativen Eingabeparadigmas selbst. Mit der Erweiterung des *Airwritings* um Steuergesten wird schließlich eine vollständige gestenbasierte Benutzerschnittstelle, genannt *Airactions*, vorgestellt.

Sowohl für das *Airwriting* als auch das *Airactions* System wurden Demonstratoren implementiert, welche die Funktionalität der entwickelten Methoden unter realistischen Bedingungen nachweisen.

9.1 Zusammenfassung der Resultate

Die Hauptbeiträge dieser Dissertation gliedern sich in die folgenden Punkte. Die Reihenfolge richtet sich nach der Abhandlung im Rahmen der Arbeit.

- Kontinuierliche Schrifterkennung mit Inertialsensoren
- Modellierung auf Einzelstrichebene
- Verfahren für den praktischen Einsatz - das *Airwriting*-System
- Gestensteuerung und Texteingabe - das *Airactions*-System

Die Hauptbeiträge werden nun noch einmal zusammengefasst und hinsichtlich ihrer Bedeutung auf die zukünftige Forschung im Bereich *Wearable Computing* diskutiert.

Kontinuierliche Schrifterkennung mit Inertialsensoren Es wird erstmalig ein vollständiges System zur Erkennung von Schrift mit Inertialsensoren, basierend auf einer Modellierung einzelner Buchstaben, beschrieben. Die Evaluation erfolgt dabei anhand von *Airwriting* Daten und auf Papier niedergeschriebener Schrift, die mit in einen Stift integrierten Inertialsensoren aufgezeichnet wurde. Damit wird sowohl für den klassischen Anwendungsfall der Online-Handschrifterkennung als auch für den neuen Fall des *Airwritings* gezeigt, dass eine Schrifterkennung auf Basis von Beschleunigungs- und Drehratensensoren möglich ist. Beide Formen der Schrifterkennung erlauben neue Anwendungen. Airwriting ermöglicht eine Texteingabe für Wearable Computer und die stiftbasierte Erkennung erlaubt die Digitalisierung traditioneller Schrift, erstmalig ohne die Verwendung von speziellem Papier oder einer externen Tracking-Technologie.

Mit dem vorgestellten Basissystem wird die Grundlage für die Erkennung von Handschrift mittels Inertialsensoren gelegt. Dies gilt sowohl für die Erkennung von *Airwriting* als auch für die Erkennung traditioneller Handschrift mittels eines Sensorstiftes. Es ist davon auszugehen, dass zukünftige Arbeiten auf dem vorgestellten System aufbauen und es weiterentwickeln werden. Das erstmalig in [AS12] beschriebene vollständige System diente bereits als Grundlage für die Arbeiten von Chen zur kontextabhängigen Modellierung der Bewegungen zwischen einzelnen Buchstaben im Rahmen seiner Dissertation [Che13] und der Arbeit von Li et al. [LYZ15], welche einen Feedbackmechanismus und ein Sprachmodell auf Buchstabenebene für die Luftschrifterkennung einführen.

Modellierung auf Einzelstrichebene Aufbauend auf der Modellierung der Buchstaben, wird in dieser Arbeit eine Modellierung auf einzelnen Strichen eingeführt und evaluiert. Aufgrund der Abhängigkeit des Beschleunigungssignals einer Bewegung von dem Kontext, in dem die Bewegung ausgeführt wird, erfolgt die Modellierung kontextabhängig. In mehreren Experimenten konnten die Vorteile einer solchen Modellierung nachgewiesen werden. Erstens konnte gezeigt werden, dass mit einer strichbasierten, kontextabhängigen Modellierung mit geringer Fehlerrate Buchstaben, Zahlen und Symbole erkannt werden können. Zweitens können Zeichen, die nicht in der Trainingsmenge vorkommen, modelliert und erkannt werden. Dies wird beispielsweise anhand der Erkennung der Ziffern und selbstdefinierter Symbole gezeigt. Drittens erlaubt die Modellierung der Einzelstriche auf einfache Art und Weise die Modellierung von Schreibvarianten. Im Falle der personenunabhängigen Erkennung von Buchstaben mit Modellierung auf Strichebene kann damit eine vergleichbare Erkennungsleistung zu einer Modellierung auf Buchstabenebene erreicht werden.

Die Beiträge zur Strichmodellierung legen den Grundstein für eine Erkennung anderer Schriftsysteme bei einer deutlichen Reduktion der nötigen Menge an Trainingsdaten. Kann für zwei Schriftsysteme eine große Schnittmenge primitiver Striche, aus denen sich die Zeichen zusammensetzen lassen, gefunden werden, so können auch die Trainingsdaten gemeinsam genutzt werden. Die Erkennung chinesischer Schriftzeichen wird beispielsweise überhaupt erst durch eine Modellierung auf Strichebene praktikabel, da bereits für den alltäglichen Bedarf 3000 bis 5000 unterschiedliche Schriftzeichen verwendet werden. Bei einer zeichenbasierten Modellierung wäre demnach im Vergleich zum lateinischen Alphabet ungefähr die 200-fache Menge an Trainingsdaten nötig.

Verfahren für den praktischen Einsatz - das *Airwriting*-System Zu den grundlegenden Beiträgen zur Schriftmodellierung und Schrifterkennung auf Basis von Inertialsensoren enthält die Arbeit auch Beiträge in Bezug auf den praktischen Einsatz als Schnittstelle. Dies beinhaltet einerseits die automatische Detektion von Schrift im kontinuierlichen Sensordatenstrom und die Komprimierung der Sensordaten, um die über Funk zu übertragende Datenmenge zu reduzieren und damit lange Einsatzzeiten zu ermöglichen. Es konnte gezeigt werden, dass sich in Alltagsaktivitäten eingebettete Schrift mit hoher Trefferquote und Genauigkeit detektieren lässt. Die Sensordaten lassen sich stark komprimieren, ohne dass damit eine große Steigerung der Fehlerrate einhergeht.

Es wurde ein Demonstrator entwickelt, welcher die Detektion und Erkennung von *Airwriting* in nahezu Echtzeit und auf einem großen Vokabular zeigt. Das System wurde vielfach ausgezeichnet und öffentlich präsentiert, unter anderem auf der CeBIT 2014. Das Interesse von Seiten der Öffentlichkeit und auch der Industrie, welches sich in zahlreichen Medienberichten und ersten Industriekooperationen zeigt, unterstreicht die praktische Anwendbarkeit und die Relevanz des vorgeschlagenen Ansatzes.

Gestensteuerung und Texteingabe - das *Airactions*-System Die Verbindung der Texteingabe via *Airwriting* mit einer Gestensteuerung (*Airactions*) wird anhand einer Beispielimplementierung einer neuartigen Benutzerschnittstelle gezeigt, welche die Erkennung von Kommandogesten und Airwriting zu einem neuen Interaktionsparadigma verbindet. Hierfür wird zusätzlich zur Detektion und Erkennung von Schrift eine Komponente zur Detektion und Erkennung von Gesten eingeführt und unter realistischen Bedingungen evaluiert. Die Integration beider Technologien erlaubt erstmalig ein freihändiges Bedienkonzept für *Wearable Computing* Anwendungen, welches über einfache Steuerkommandos hinausgeht. Mit diesem Funktionsumfang lassen sich bereits alle wesentlichen Bedienelemente moderner Smartphones und Smartwatches abbilden.

Damit eröffnen sich viele Möglichkeiten für weiterführende Arbeiten, da erstmalig eine vollständige, gestenbasierte und mobile Benutzerschnittstelle existiert, die nun in unterschiedlichen Anwendungsszenarien eingesetzt und evaluiert werden kann. Da das *Airactions*-System alle wesentlichen Anforderungen an Benutzerschnittstellen im Bereich des *Wearable Computings* erfüllt, ist davon auszugehen, dass es eine wichtige Rolle in der weiteren Forschung und auch praktischen Anwendung einnehmen wird.

9.2 Weiterführende Arbeiten

Naturgemäß wirft die vorliegende Arbeit neue Forschungsfragen auf, da mit *Airwriting* Neuland betreten wurde. Einige dieser Fragen werden hier benannt und liefern Anstöße für mögliche zukünftige Forschung auf dem Gebiet der Schrifterkennung mit Inertialsensoren und der Anwendung im Bereich des *Wearable Computing*. Es lassen sich zwei Bereiche unterscheiden, in denen sich interessante Fragestellungen ergeben. Zum Ersten ist dies der technische Bereich, indem es primär um eine Verbesserung der Fehlerraten in verschiedenen Anwendungskontexten geht. Zum Zweiten ist das der Bereich

der Mensch-Maschine-Interaktion, indem es um die Fragestellungen geht, wie gut die Technik als Schnittstelle funktioniert und wie Menschen diese Schnittstelle tatsächlich nutzen.

Für die technische Verbesserung des Systems seien beispielhaft einige relevante Punkte genannt, die auch im praktischen Einsatz des Systems von großer Bedeutung wären. Da im Wearable Computing in der Regel das Eingabegerät ein persönliches Artefakt des Nutzers ist, ist es naheliegend die Erkennung über die Nutzungsdauer an die Präferenzen und Charakteristika des Nutzers anzupassen. In der automatischen Spracherkennung wurden unterschiedliche Methoden für eine automatische Sprecheradaption zur Laufzeit entwickelt. Diese Methoden könnten auf den vorliegenden Fall der Schrifterkennung übertragen werden, um die personenspezifischen Unterschiede in der Erkennungsleistung zu reduzieren.

Im Kontext der Modellierung von Einzelstrichen stellt sich die Frage, ob eine optimale Menge von Universaleinheiten gefunden werden kann, also eine Art Basisalphabet von Bewegungen, aus dem sich beliebige Zeichen aus beliebigen Schriftsystemen zusammensetzen lassen. Auch der konkrete Transfer, beispielsweise auf chinesische Schriftzeichen, mit den in dieser Arbeit vorgestellten Methoden wäre ein weiterer Schritt.

Weiterhin wäre eine Analyse und Verbesserung der Fehlerrate unter realen Bedingungen interessant. Eine Nutzung während des Gehens oder während anderer Aktivitäten macht die Erkennung von Schrift durch die Überlagerung mit anderen Körperbewegungen beispielsweise vermutlich schwieriger.

Ausführliche Usability Studien des *Airwriting*- und *Airactions*-Systems, mit einer großen und diversen Probandengruppe unter realistischen Nutzungsbedingungen, fanden im Rahmen dieser Arbeit noch nicht statt. Hier ergeben sich eine ganze Reihe weiterführender Fragestellungen. Häufig muss für deren Beantwortung erst das System selbst in genügend guter Qualität zur Verfügung stehen. Dies ist nun mit den Ergebnissen dieser Arbeit gewährleistet. Die Forschung im Bereich der Texteingabe hat über die letzten Jahre ein striktes Rahmenwerk zur Evaluation von Texteingabemethoden hinsichtlich ihrer Effizienz entwickelt. In der Regel erfordert dies einen Korrekturmechanismus, der Nutzern erlaubt Fehler zu korrigieren. Nach Implementierung eines geeigneten Mechanismus wäre ein direkter Vergleich mit anderen Texteingabemethoden für mobile Geräte wie beispielsweise Softtastaturen, Geräten wie dem Twiddler oder Gestentastaturen möglich. Für diese Methoden wurden in der Regel entsprechende Untersuchungen bereits durchgeführt. Ein weiterer relevanter Punkt ist die Frage, wie sich die Abwesenheit visueller Aufmerksamkeit auf die Schrift und damit deren Erkennung auswirkt.

Im Idealfall könnten Nutzer „blind“ schreiben. Letztlich wäre in Benutzerstudien zu klären, ob und unter welchen Bedingungen Nutzer gestenbasierte Handschrift zur Texteingabe gegenüber anderen Methoden, wie Softtastaturen oder auch der Spracherkennung, präferieren. Der Beantwortung dieser Fragen wird im Rahmen einer geplanten Kommerzialisierung der Technologie nachgegangen werden.

Literaturverzeichnis

[AAS+09] Oliver Amft, Roman Amstutz, Asim Smailagic, Dan Siewiorek, and Gerhard Tröster. Gesture-Controlled User Input to Complete Questionnaires on Wrist-Worn Watches. In *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, pages 131–140. Springer-Verlag, 2009.

[ACG+11] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. Using mobile phones to write in air. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2011.

[AGS10] Christoph Amma, Dirk Gehrig, and Tanja Schultz. Airwriting recognition using wearable motion sensors. In *Proceedings of the 1st Augmented Human International Conference*. ACM, 2010.

[AMG07] Thierry Artieres, Sanparith Marukatat, and Patrick Gallinari. Online handwritten shape recognition using segmental hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):205–217, 2007.

[Amm09] Christoph Amma. Airwriting recognition using wearable motion sensors. Diplomarbeit, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2009.

[AS12] Christoph Amma and Tanja Schultz. Airwriting: demonstrating mobile text input by 3d-space handwriting. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 319–320. ACM, 2012.

[B+06] Christopher M Bishop et al. *Pattern recognition and machine learning*. Springer, New York, 2006.

[BCK+03] Won-Chul Bang, Wook Chang, Kyeong-Ho Kang, Eun-Seok Choi, Alexey Potanin, and Dong-Yoon Kim. Self-contained spatial input device for wearable computers. In *Proceedings of the 16th International Symposium on Wearable Computers*, pages 26–26. IEEE, 2003.

[BMB12] Bartosz Bajer, I Scott MacKenzie, and Melanie Baljko. Huffman Base-4 Text Entry Glove (H4 TEG). In *Proceedings of the 16th International Symposium on Wearable Computers*, pages 41–47. IEEE, 2012.

[BSLJ03] H Brashear, T Starner, P Lukowicz, and H Junker. Using multiple sensors for mobile sign language recognition. In *Proceedings of the 7th International Symposium on Wearable Computers*. IEEE, 2003.

[Bur98] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[CABL13] Jingyuan Cheng, Oliver Amft, Gernot Bahle, and Paul Lukowicz. Designing sensitive wearable capacitive sensors for activity recognition. *IEEE Sensors Journal*, 13(10):3935–3947, 2013.

[CAJ13] Mingyu Chen, Ghassan AlRegib, and Biing-Hwang Juang. Feature processing and modeling for 6d motion gesture recognition. *IEEE Transactions on Multimedia*, 15(3):561–571, 2013.

[Che13] Mingyu Chen. *Universal Motion-Based Control and Motion Recognition*. PhD thesis, Georgia Institute of Technology, 2013.

[CK04] Sung-Jung Cho and Jin H Kim. Bayesian network modeling of strokes and their relationships for on-line handwriting recognition. *Pattern Recognition*, 37(2):253–264, 2004.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[DKHR14] T. Deselaers, D. Keysers, J. Hosang, and H.A. Rowley. GyroPen: Gyroscopes for Pen-Input With Mobile Phones. *IEEE Transactions on Human-Machine Systems*, PP(99):1–9, 2014.

[DP09] Moussa Djioua and Rejean Plamondon. A new algorithm and system for the characterization of handwriting strokes with delta-lognormal parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2060–2072, 2009.

[EM02] Y. Ephraim and N. Merhav. Hidden Markov processes. *IEEE Transactions on Information Theory*, 48(6):1518 –1569, jun 2002.

[GR93] David Goldberg and Cate Richardson. Touch-typing with a stylus. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 80–87. ACM, 1993.

[HAH01] X. Huang, A. Acero, and H.W. Hon. *Spoken language processing*. Prentice Hall, 2001.

[HHH98] Frank G Hofmann, Peter Heyer, and Günter Hommel. Velocity profile based recognition of dynamic gestures with discrete hidden markov models. In *Gesture and Sign Language in Human-Computer Interaction*, pages 81–95. Springer, 1998.

[HLB00] Jianying Hu, Sok Gek Lim, and Michael K Brown. Writer independent on-line handwriting recognition using an hmm approach. *Pattern Recognition*, 33(1):133–147, 2000.

[JALT08] Holger Junker, Oliver Amft, Paul Lukowicz, and Gerhard Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6):2010–2024, 2008.

[JR90] Biing-Hwang Juang and Lawrence Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(9):1639–1641, 1990.

[KCHP01] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings IEEE International Conference on Data Mining*, pages 289–296. IEEE, 2001.

[KCK06] Dae Hwan Kim, Hyun Il Choi, and Jin Hyung Kim. 3d space handwriting recognition with ligature model. In *Ubiquitous Computing Systems*, pages 41–56. Springer, 2006.

[KD14] Andreas Komninos and Mark Dunlop. Text input on a smart watch. *Pervasive Computing*, 13(4):50–58, 2014.

[KNQ12] Per Ola Kristensson, Thomas Nicholson, and Aaron Quigley. Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors. In *Proceedings*

of the 2012 ACM international conference on Intelligent User Interfaces, pages 89–92. ACM, 2012.

[Koe03] Alessandro L Koerich. Unconstrained handwritten character recognition using different classification strategies. In *International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR)*, 2003.

[KZ04] Per-Ola Kristensson and Shumin Zhai. SHARK 2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 43–52. ACM, 2004.

[Lee88] Kai-Fu Lee. *Automatic speech recognition: the development of the SPHINX system*, volume 62. Springer Science & Business Media, 1988.

[LK99] Hyeon-Kyu Lee and Jin-Hyung Kim. An HMM-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973, 1999.

[LLJ06] Yang Liu, Xiabi Liu, and Yunde Jia. Hand-gesture based text input for wearable computers. In *International Conference on Computer Vision Systems*, pages 8–8. IEEE, 2006.

[LPS04] Kent Lyons, Daniel Plaisted, and Thad Starner. Expert chording text entry on the twiddler one-handed keyboard. In *Proceedings of the 8th International Symposium on Wearable Computers*, pages 94–101. IEEE, 2004.

[LSP+04] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and EW Looney. Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 671–678. ACM, 2004.

[LYZ15] Yujia Li, Kaisheng Yao, and Geoffrey Zweig. Feedback-Based Handwriting Recognition from Inertial Sensor Data for Wearable Devices. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2015.

[LZWV09] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave: Accelerometer-based personalized gesture reco-

gnition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[MA07] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, 2007.

[Man97] Steve Mann. Wearable computing: A first step toward personal imaging. *Computer*, 30(2):25–32, 1997.

[MBS93] Thomas M Martinetz, Stanislav G Berkovich, and Klaus J Schulten. Neural-gas' network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, 4(4):558–569, 1993.

[MHV11] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *IEEE International Conference on Rehabilitation Robotics*, pages 1–7. IEEE, 2011.

[MJH14] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. Vulture: a mid-air word-gesture keyboard. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1073–1082. ACM, 2014.

[MM95] N. Mai and C. Marquardt. Analyse und therapie motorischer schreibstörungen. *Psychologische Beiträge*, 37:538–582, 1995.

[MMB96] Edgar Matias, I Scott MacKenzie, and William Buxton. One-handed touch typing on a QWERTY keyboard. *Human-Computer Interaction*, 11(1):1–27, 1996.

[MMC09] Pranav Mistry, Pattie Maes, and Liyan Chang. Wuw-wear ur world: a wearable gestural interface. In *CHI'09 extended abstracts on Human factors in computing systems*, pages 4111–4116. ACM, 2009.

[Mor81] Pietro Morasso. Spatial control of arm movements. *Experimental brain research*, 42(2):223–227, 1981.

[Mor11] Daniel Morlock. Gestenbasierte Texteingabe an großen Displays. Diplomarbeit, Karlsruhe Institue of Technology, 2011.

[NBN11] Tao Ni, Doug Bowman, and Chris North. Airstroke: bringing unistroke text entry to freehand gesture interfaces. In *Procee-*

dings of the SIGCHI Conference on Human Factors in Computing Systems, pages 2473–2476. ACM, 2011.

[NS03] Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146. ACM, 2003.

[OHOW13] Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. ZoomBoard: a diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2799–2802. ACM, 2013.

[OVWY94] JJ Odell, V. Valtchev, PC Woodland, and SJ Young. A one pass decoder design for large vocabulary recognition. In *Proceedings of the workshop on Human Language Technology*, pages 405–410. Association for Computational Linguistics, 1994.

[PLH+11] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 260–273. ACM, 2011.

[Pre08] Zoltán Prekopcsák. Accelerometer based real-time gesture recognition. In *Proceedings of the 12th International Student Conference on Electrical Engineering*, 2008.

[PS00] Réjean Plamondon and Sargur N Srihari. Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, 2000.

[Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RB10] Julie Rico and Stephen Brewster. Usable gestures for mobile interfaces: evaluating social acceptability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 887–896. ACM, 2010.

[Rek01] Jun Rekimoto. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Proceedings of the 5th International Symposium on Wearable Computers*, pages 21–27. IEEE, 2001.

[RLNS10] Giuseppe Raffa, Jinwon Lee, Lama Nachman, and Junehwa Song. Don't slow me down: Bringing energy efficiency to continuous gesture recognition. In *ISWC*, pages 1–8, 2010.

[RS99] Robert Rosenberg and Mel Slater. The chording glove: a glove-based text input device. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 29(2):186–191, 1999.

[Sch02] Tanja Schultz. GlobalPhone: A multilingual speech and text database developed at Karlsruhe University. In *Proceedings of the International Conference on Spoken Language Processing*, 2002.

[SL81] JF Soechting and F Lacquaniti. Invariant characteristics of a pointing movement in man. *The Journal of Neuroscience*, 1(7):710–720, 1981.

[SMA+12] Alexander Schick, Daniel Morlock, Christoph Amma, Tanja Schultz, and Rainer Stiefelhagen. Vision-based handwriting recognition for unrestricted text input in mid-air. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 217–220. ACM, 2012.

[SMFW01] H. Soltau, F. Metze, C. Fügen, and A. Waibel. A one-pass decoder based on polymorphic linguistic context assignment. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2001.

[SMR+97] Thad Starner, Steve Mann, Bradley Rhodes, Jeffrey Levine, Jennifer Healey, Dana Kirsch, Rosalind W Picard, and Alex Pentland. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6(4):386–398, 1997.

[SP97] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer, 1997.

[SPHB08] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In *Proceedings*

of the 2nd international conference on Tangible and embedded interaction, pages 11–14. ACM, 2008.

[SS09] Emilian Stoimenov and Tanja Schultz. A multiplatform speech recognition decoder based on weighted finite-state transducers. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 293–298. IEEE, 2009.

[Sta13] Thad Starner. Project glass: An extension of the self. *Pervasive Computing*, 12(2):14–16, 2013.

[Sta14] Thad Starner. How wearables worked their way into the mainstream. *Pervasive Computing*, 13(4):10–15, 2014.

[TWG+14] Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze, Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt, Tim Schlippe, Matthias Janke, et al. BioKIT-Real-time decoder for biosignal processing. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association*, 2014.

[VT80] Paolo Viviani and Carlo Terzuolo. Space-time invariance in learned motor skills. *Advances in psychology*, 1:525–533, 1980.

[WCB11] Julie R Wiliamson, Andrew Crossan, and Stephen Brewster. Multimodal mobile interactions: usability studies in real world settings. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 361–368. ACM, 2011.

[WH99] Ying Wu and Thomas S Huang. Vision-based gesture recognition: A review. In *Gesture-based communication in human-computer interaction*, pages 103–115. Springer, 1999.

[Win14] Fabian Winnen. Cloudgestützte Gestenerkennung mittels eines Sensorarmbandes zur Steuerung Android-basierter Geräte. Bachelorarbeit, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014.

[Woo07] Oliver J. Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, 2007.

[YOI92] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385. IEEE, 1992.

[You08] Steve Young. HMMs and related speech recognition technologies. In *Springer Handbook of Speech Processing*, pages 539–558. Springer, 2008.

Anhang A

A.1 Strichsequenzen

Variante	Sequenz Primitive
0	$K_{lu}K_{ro}$
1	$G_{or}G_{u}$
2	$K_{or}G_{ul}G_{r}$
3	$K_{ru}K_{ru}$
4	$G_{ul}G_{r}G_{o}G_{u}$
5	$G_{u}K_{ru}G_{o}G_{r}$
6	$K_{lu}K_{ro}$
7	$G_{r}G_{ul}$
8	$K_{lu}K_{ru}K_{lo}K_{ro}$
9	$K_{lu}G_{or}G_{u}K_{ul}$

Tabelle A.1 – Sequenzen primitiver Striche für den Strich.Z-Datensatz.

Variante	Sequenz Primitive
Dreieck	$G_{ul}G_rG_{ol}$
Epsilon	$G_lG_{ur}G_{ul}G_r$
S gespiegelt	$K_{ru}K_{lu}$
U-Dach	$K_{ur}G_l$
U-Dach gespiegelt	$K_{ul}G_r$
Viereck	$G_uG_rG_oG_l$
Dreieck gespiegelt	$G_{ur}G_{or}G_l$
Schlange	$K_{or}K_{ur}$
Schlange gespiegelt	$K_{ur}K_{or}$

Tabelle A.2 – Sequenzen primitiver Striche für den Strich.S-Datensatz. Die Reihenfolge entspricht Abbildung 3.3(e) von links nach rechts.

Variante	Sequenz Primitive
A	$G_{ul}G_{or}G_{ur}G_{ol}G_r$
B	$G_uG_oK_{ru}K_{ru}$
C	K_{lu}
D	$G_uG_oK_{ru}$
E	$G_uG_rG_{ol}G_rG_{ol}G_r$
F	$G_uG_{ol}G_rG_{ul}G_r$
G	$K_{lu}G_{or}G_uG_{ol}$
H	$G_uG_{or}G_uG_{ol}G_r$
I	G_u
J	G_uK_{ul}
K	$G_uG_{or}G_{ul}G_{ur}$
L	G_uG_r
M	$G_uG_oG_{ur}G_{or}G_u$
N	$G_oG_{ur}G_o$
O	$K_{lu}K_{ro}$
P	$G_uG_oK_{ru}$
Q	$K_{lu}K_{ro}K_{lu}$
R	$G_uG_oK_{ru}G_{ur}$
S	$K_{lu}K_{ru}$
T	$G_uG_{ol}G_r$
U	K_{ur}
V	$G_{ur}G_{or}$
W	$G_{ur}G_{or}G_{ur}G_{or}$
X	$G_{ur}G_oG_{ul}$
Y	$G_{ur}G_{or}G_{ul}$
Z	$G_rG_{ul}G_r$

Tabelle A.3 – Sequenzen primitiver Striche für den Strich.B-Datensatz.

Variante	Sequenz Primitive	Variante	Sequenz Primitive
A	$G_{or}G_{ur}G_{ol}G_r$	K	$G_uG_{or}G_{ul}G_{ur}$
A(2)	$G_{or}G_{ur}G_{ol}$	K(2)	$G_uG_{or}G_{ul}K_{lo}G_{ur}$
A(3)	$G_{ul}G_{or}G_{ur}G_{ol}G_r$	L	G_uG_r
B	$G_uG_oK_{ru}K_{ru}$	M	$G_uG_oG_{ur}G_{or}G_u$
C	K_{lu}	M(2)	$G_oG_{ur}G_{or}G_u$
D	$G_uG_oK_{ru}$	N	$G_oG_{ur}G_o$
E	$G_uG_rG_{ol}G_rG_{ol}G_r$	N(2)	$G_uG_oG_{ur}G_o$
E(2)	$G_uG_oG_rG_{ul}G_rG_{ul}G_r$	O	$K_{lu}K_{ro}$
E(3)	$G_uG_rG_lG_{or}G_lG_{or}G_l$	P	$G_uG_oK_{ru}$
E(4)	$G_lG_oG_rG_{ol}G_r$	Q	$K_{lu}K_{ro}G_uG_{ur}$
E(5)	$K_{lu}G_{ol}G_r$	R	$G_uG_oK_{ru}G_{ur}$
F	$G_uG_{ol}G_rG_{ul}G_r$	S	$K_{lu}K_{ru}$
F(2)	$G_uG_{or}G_lG_{ur}G_l$	T	$G_uG_{ol}G_r$
F(3)	$G_uG_oG_rG_{ul}G_r$	T(2)	$G_uG_{or}G_l$
G	$K_{lu}G_uG_{or}G_l$	U	K_{ur}
G(2)	$K_{lu}G_oG_rG_u$	U(2)	$G_rK_{ur}G_uG_r$
G(3)	$K_{lu}G_oG_l$	U(3)	$K_{ur}G_uG_r$
G(4)	$K_{lu}G_oG_lG_rG_u$	V	$G_{ur}G_{or}$
G(5)	$K_{lu}G_{ol}G_rG_u$	W	$G_{ur}G_{or}G_{ur}G_{or}$
H	$G_uG_{or}G_uG_{ol}G_r$	X	$G_{ur}G_oG_{ul}$
H(2)	$G_uG_{or}G_uG_oG_l$	X(2)	$G_{ur}G_lG_{or}$
I	G_u	X(3)	$G_{ul}G_oG_{ur}$
I(2)	$G_uG_rG_lG_{or}G_l$	Y	$G_{ur}G_{or}G_{ul}$
J	$G_rG_uK_{ul}$	Y(2)	$G_{ur}G_{or}G_{ul}G_u$
J(2)	$G_rG_uK_{ul}G_{or}$	Y(3)	$G_{ur}G_uG_oG_{or}$
J(3)	$G_oK_{ur}G_uK_{ul}$	Z	$G_rG_{ul}G_rG_{ol}G_r$
J(4)	$G_oK_{ur}G_uK_{ul}G_{or}$	Z(2)	$G_rG_{ul}G_r$
J(5)	G_uG_l		
J(6)	G_uK_{ul}		

Tabelle A.4 – Schreibvarianten der Buchstaben mit entsprechender Sequenz von Primitiven.