



Universität  
Potsdam  
Institut für Informatik  
Lehrstuhl für Komplexe Multimediale Anwendungsarchitekturen

Bachelorarbeit

# **Untersuchung und exemplarische Evaluierung der Möglichkeiten zur Gestenerkennung durch mobile Nutzerendgeräte**

- Potsdam - 30. August 2011 -

von

Linda Pfeiffer geb. Tschepe (tschepe@uni-potsdam.de), Matrikel-Nr.: 738190

1. Gutachter: Prof. Dr.-Ing. Ulrike Lucke (ulrike.lucke@uni-potsdam.de)
2. Gutachter: Prof. Dr. Torsten Schaub (torsten@cs.uni-potsdam.de)
- Betreuer: Dr.-Ing. Raphael Zender (zender@uni-potsdam.de)

## **Zusammenfassung**

Mit der Einführung der aktuellen Klasse mobiler Nutzerendgeräte (z.B. Smartphones, Tablet- PCs) entstanden eine Reihe darauf zugeschnittener Konzepte für die Interaktion zwischen Mensch und Maschine. Diese fokussieren die Bedienung über berührungs-empfindliche, kompakte Displays ohne zusätzliche Hilfsmittel wie Stifte oder Tastaturen. Auch die Auswertung von Bewegungsinformationen ist dank integrierter Beschleunigungssensoren sowie Gyroskopen möglich und hat sich vor allem im Bereich Mobile Games etabliert. Diese Bachelorarbeit untersucht aktuelle Software-Lösungen, mit denen derartige Informationen ausgelesen und weiterverarbeitet werden können. Im Fokus steht dabei die Gestenerkennung durch Smartphones oder vergleichbare Geräteklassen. Die existierenden Systeme und Konzepte werden miteinander verglichen und hinsichtlich ihrer Eignung für ausgewählte Anwendungsbereiche bewertet. Weiterhin wird ein konkretes Gestenerkennungssystem ausgewählt. Die Rahmenbedingungen seiner Nutzung werden ermittelt und durch einen exemplarischen Einsatz wird die Nutzbarkeit des Systems überprüft.

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>7</b>
<b>2. Grundlagen</b>	<b>9</b>
2.1. Herausforderungen . . . . .	9
2.2. Sensorik . . . . .	10
2.3. Verwandte Arbeiten . . . . .	12
2.4. Algorithmen . . . . .	13
2.4.1. Hidden Markov Modelle (HMMs) . . . . .	13
2.4.2. Dynamic Time Warping (DTW) . . . . .	14
2.4.3. Support Vector Machine (SVM) . . . . .	15
2.4.4. K-Nearest-Neighbors (kNN) . . . . .	15
2.4.5. K-Means . . . . .	16
2.4.6. Entscheidungsbäume . . . . .	16
2.5. Zusammenfassung . . . . .	17
<b>3. Marktrecherche</b>	<b>18</b>
3.1. Zusammenfassung . . . . .	21
<b>4. Implementierung</b>	<b>22</b>
4.1. Beschreibung des Einsatzfeldes . . . . .	22
4.2. Auswahl der Gesten . . . . .	23
4.3. Auswahl des Systems/ der App . . . . .	26
4.4. Aufsetzen des Systems . . . . .	28
4.5. Evaluation . . . . .	32
4.6. Zusammenfassung . . . . .	32
<b>5. Fazit und Ausblick</b>	<b>34</b>

<b>Literaturverzeichnis</b>	<b>36</b>
<b>Internetquellen</b>	<b>39</b>
<b>A. Erklärung</b>	<b>41</b>
<b>B. Source Code</b>	<b>42</b>

# Abbildungsverzeichnis

2.1.	Prototyp des Imaginary Interfaces in Benutzung: Die Kamera ist bequem an der Kleidung am Torso befestigt.(Quelle: [GBB10]) . . . . .	10
2.2.	HMM mit drei versteckten Zuständen, deren Start- und Übergangswahr- scheinlichkeiten ( $\pi_i, a_{ij}$ ), sowie drei Beobachtungen $O_t$ und deren Auftritts- wahrscheinlichkeiten $b_{it}$ . . . . .	14
2.3.	Dynamic Time Warping - Visualisierung . . . . .	15
2.4.	eine Soft-Margin SVM . . . . .	16
2.5.	Entscheidungsbaum, wie in [PEK <sup>+</sup> 06] verwendet . . . . .	17
4.1.	Einsammeln einer Batterie - Gestenauswahl . . . . .	25
4.2.	Versenden einer Nachricht - Gestenauswahl . . . . .	25
4.3.	Kommunikation im System . . . . .	29
4.4.	Interface der Android-App zur Dateneingabe . . . . .	30
4.5.	Struktur der Android-App . . . . .	30
4.6.	Erweiterung der App 'Motion Gesture Ad' . . . . .	31

# Tabellenverzeichnis

3.1. Überblick über aktuelle Apps . . . . .	19
4.1. Systeme zur Erkennung komplexer Gesten . . . . .	27
4.2. Erkennungsrate des Systems . . . . .	32

# 1. Einleitung

Smartphones und ähnliche Geräte sind heute zum ständigen Begleiter des Menschen geworden und ausgestattet mit Internetzugang und allerlei Sensoren werden sie auch immer wieder als kleine 'Allesköninger' bezeichnet. Die Interaktionsformen konzentrieren sich hauptsächlich auf berührungsempfindliche Displays, die auf Eingaben mit dem Finger oder Stift reagieren. Aber auch Sprach- und Gestenerkennung werden immer häufiger genutzt. Letzteres ist insbesondere aus den Bereichen Unterhaltung und Spiele nicht mehr wegzudenken. Seit der Einführung der stationären Spielekonsole Wii 2006 ist die Eingabe durch Bewegung enorm populär. Dieser Trend zeichnet sich nun auch für Spiele auf mobilen Geräten ab.

Doch inwieweit geht heute die Verwendung von Gesten als Eingabemethode über den Unterhaltungsbereich hinaus? Immerhin könnte die Gestenerkennung in naher Zukunft, insbesondere in Kombination mit Spracherkennung, der Schlüssel zu intuitiven effizienten und natürlichen Benutzerschnittstellen sein, die relativ umgebungsunabhängig an vielen Orten zum Einsatz kommen können. Dies röhrt daher, dass der Mensch von Natur aus fast ausschließlich über Sprache, Gestik und Mimik kommuniziert. Doch sind heutige Konzepte und Algorithmen schon ausgereift genug, um eine zuverlässige und sinnvolle Schnittstelle zum Benutzer zu bilden?

Diese Bachelorarbeit untersucht aktuelle Software-Lösungen, die Bewegungsinformationen auslesen und weiterverarbeiten. Im Fokus steht dabei die Erkennung von Hand- und Armgesten, da diese, gegenüber Gesten anderer Körperteile, über eine hohe Aussagekraft verfügen und in der menschlichen Kommunikation häufiger zum Einsatz kommen [MA07]. Im Bezug auf die Hardware steht die Gestenerkennung durch Smartphones oder vergleichbare Gerätetypen, welche bereits ein hohes Maß an Vertrautheit genießen, im Mittelpunkt.

Das zweite Kapitel der Arbeit befasst sich zunächst mit den Grundlagen der Gestenerkennung auf mobilen Nutzerendgeräten. Es wird der Einsatz diverser Sensoren und Algorithmen diskutiert und es werden spezielle Herausforderungen beim Entwurf eines Gestenerkennungssystems herausgestellt. Darauf folgend (Kapitel 3) werden die Ergebnisse detailliert vorgestellt.

nisse der Recherche über heutige Gestenerkennungssysteme für mobile Nutzerendgeräte präsentiert. Diese bedienen auch heute schon vielfältige Einsatzgebiete und können nach der Art der erfassten Bewegung unterteilt werden. Das vierte Kapitel behandelt den exemplarischen Einsatz eines der recherchierten Systeme für ein mobiles pervasives Lernspiel. Dies umfasst nicht nur die sorgfältige Auswahl des Systems, sondern auch die Bestimmung intuitiver Gesten zur Interaktion, die Anpassung des Systems an die Gegebenheiten und Anforderungen des Einsatzszenarios und eine Bewertung der Nutzbarkeit des Systems für den Einsatzzweck. Eine finale Zusammenfassung der Ergebnisse dieser Arbeit sowie ein Ausblick über eventuell anschließende Arbeiten ist in Kapitel fünf zu finden.

## 2. Grundlagen

In den letzten Jahren war die Gestenerkennung auf mobilen Nutzerendgeräten immer wieder Thema diverser Forschungsarbeiten, welche die Thematik vorantreiben. Bevor ein Blick auf einige dieser Arbeiten geworfen wird, erläutert dieses Kapitel zunächst einige Herausforderungen, die beim Entwurf eines solchen Systems bestehen, und untersucht verschiedene Sensoren in heutigen mobilen Geräten, mit denen Bewegungsdaten erfasst werden können, auf ihre Vor- und Nachteile hin. Der letzte Teil des Kapitels beinhaltet dann eine genauere Betrachtung bisher verwendeter Algorithmen.

### 2.1. Herausforderungen

Für die Gestaltung von Gestenerkennungssystemen sind zunächst die selben Punkte wichtig, wie für die Gestaltung jeder anderen Benutzerschnittstelle. Laut Kortum sind in Teil 11 der ISO 9241 die Eigenschaften Effektivität, Effizienz und Nutzerzufriedenheit hervorgehoben [Kor08]. Der Nutzer soll demnach sein Ziel erreichen und dies mit möglichst geringem Aufwand und möglichst komfortabel. Folglich ist bei der Gestaltung eines Gestenerkennungssystems viel Aufmerksamkeit dem Design des Gestenalphabets zu widmen. Dieses sollte in Umfang und Art der Gesten so gestaltet sein, dass eine zuverlässige Erkennung möglich ist. Die Gesten sollten sich deswegen klar voneinander unterscheiden. Um dem Nutzer eine zügige und zufriedenstellende Interaktion bieten zu können sollten die Gesten intuitiv und natürlich sein. Die Algorithmik eines solchen Systems muss mit der räumlichen und zeitlichen Variabilität der Gesten umzugehen verstehen. Ein Mensch besitzt nicht die Fähigkeit eine Geste zweimal exakt gleich auszuführen. Diese Variabilität nimmt zu, wenn die gleiche Geste von verschiedenen Nutzern ausgeführt wird [MA07]. Dennoch ist es wünschenswert, ein persönliches Training des System durch den Nutzer zu umgehen. Dies könnte als wenig komfortabel empfunden werden, was zu einer ablehnenden Haltung gegenüber dem System führen könnte. Neben den Herausforderungen, die im Allgemeinen bei Gestenerkennungssystemen vorliegen, muss bei mobilen Systemen bei der Auswahl der Komponenten insbesondere auf die Ressourcenbeschränktheit

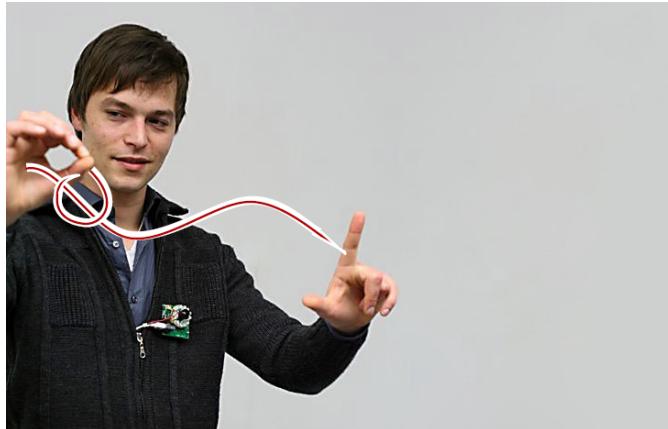


Abbildung 2.1.: Prototyp des Imaginary Interfaces in Benutzung: Die Kamera ist bequem an der Kleidung am Torso befestigt.(Quelle: [GBB10])

mobiler Geräte im Vergleich zu stationären Computern geachtet werden [Nie08]. Die Algorithmen und Modelle zur Gestenerkennung sollten so gewählt werden, dass das System trotz dieser Beschränkungen nahezu in Echtzeit reagieren kann. Dabei bildet jedoch der Anwendungsteil zur Erkennung der Gesten lediglich die Schnittstelle zum Menschen und der Großteil der Rechenzeit sollte anderen Komponenten vorbehalten bleiben. Bei der Wahl der Geräte und der Sensorik ist bei mobilen Geräten insbesondere der Energieverbrauch zu beachten [Nie08], da der Energiespeicher des Geräts eine möglichst lange Betriebsdauer bieten soll.

## 2.2. Sensorik

Für die Erfassung von Bewegungsdaten bieten sich bei aktuellen mobilen Geräten im Wesentlichen zwei Möglichkeiten an:

- die Erfassung der Bewegung durch eine Kamera,
- die Erfassung der Daten durch Sensoren, wie Accelerometer und Gyroskope.

Mobile Bild-basierte Systeme könnten beispielsweise eine am Torso oder Kopf des Nutzers befestigte Kamera nutzen, um mit ihr Gesten der Hände zu erfassen. So werden zum Beispiel bei den von Gustafson et al. entwickelten Imaginary Interfaces, die Handgesten zur Kommunikation mit den bildschirmlosen Geräten erfasst [GBB10] (Siehe Abbildung 2.1). Es ist aber auch die Nutzung von Kameras vorstellbar, die in Geräte, wie Smart-

phones, integriert sind. Während das Gerät bewegt wird, kann aus der Verschiebung des Umgebungsbildes auf die Bewegung des Gerätes zurück geschlossen und somit wiederum Handgesten erfasst werden.

Accelerometer (auch Beschleunigungssensor oder G-Sensor genannt) messen die Beschleunigung des Geräts, indem sie die Trägheitskraft bestimmen, die auf eine Testmasse wirkt. Gyroskope (oder auch Kreiselinstrument, Gyrosensor oder Gyrometer) messen Drehbewegungen und erkennen durch das Verhältnis zur Ausgangslage die aktuelle Neigung bzw. Lage des Geräts. Moderne Smartphones besitzen inzwischen nahezu alle ein Beschleunigungssensor und einige Geräte, wie z.B. das iPhone 4 [18], enthalten zusätzlich auch ein Gyroskop.

Auch wenn Kameraaufzeichnungen traditionell zur Gestenerkennung genutzt werden, wird für die Erkennung von Bewegung auf mobilen Geräten zumeist die Verwendung von Sensoren bevorzugt. Dies kommt daher, dass die Verwendung von Kameras adäquate Lichtverhältnisse voraussetzt, dass eine Kamera mehr Energie verbraucht als ein Accelerometer und daher, dass die benötigte Rechenleistung bei der Verwendung von Sensoren oft geringer ist, da die Bewegungsdaten bei Sensormessungen direkt vorliegen und nicht erst aus dem Bildmaterial heraus gerechnet werden müssen [HHK09], [Nie08].

Die meisten heutiger Systeme arbeiten mit dem Built-In-Accelerometer der Smartphones und, wenn zur Verfügung, einem Gyrosensor (Siehe auch Tabelle 4.1). Es besteht die Möglichkeit durch die Verwendung von mehreren Sensoren die Erkennungsrate der Systeme zu erhöhen, jedoch bleibt dabei zu bedenken, dass mit der Zunahme der Sensoren auch die Menge der Daten erheblich steigt, was zu einem deutlich erhöhten Rechenaufwand führen würde [LD10], [Nie08]. In den meisten Fällen lohnen sich die Verbesserungen der Erkennungsrate im Verhältnis zur Zunahme der Rechenlast auf den doch recht beschränkten mobilen Geräten nicht.

Bevor jedoch die Gedanken auf die Verwendung von zusätzlichen Sensoren gerichtet werden, bleibt noch die Frage, ob die Messbereiche und Abtastraten der in heutigen mobilen Geräten verwendeten G-Sensoren für die Gestenerkennung ausreichend sind. Hein et al. leiteten aus eigenen Tests wie auch aus fremden Arbeiten einen Messbereich von  $+/- 4g$  und eine Abtastrate von 32 Hz als sinnvoll zur Erfassung von menschlichen Handgesten ab [HHK09]. Für eine Aktivitätenerkennung können hingegen schon Samplingraten von 20 Hz ausreichend sein [LD10]. Mit dem Sensor des iPhones sollen beispielsweise nach [Kli09] ein Messbereich von etwa  $+/- 2g$  und eine Abtastrate von bis zu 100Hz möglich sein. Der von Hein et al. bevorzugte Messbereich von  $+/- 4g$  ist also nicht unbedingt vorhanden, doch konnten diverse der Arbeiten im folgenden Abschnitt, wie [CCB<sup>+</sup>06] und [CMC10], auch mit diesem Messbereich gute Ergebnisse erzielen.

## 2.3. Verwandte Arbeiten

Diverse Forschungsarbeiten umfassten die Entwicklung eines Systems, mit welchem Bewegungsdaten auf mobilen Geräten verarbeitet werden können. Dabei wurde häufig auf Smartphones und deren Built-In-Sensorik zurückgegriffen. Diese Geräte haben den Vorteil, dass sie sich bereits einer großen Verbreitung erfreuen und sich eventuelle Berührungsängste von Nutzern gegenüber neuen Technologien in Grenzen halten. Nachteilig an der Verwendung dieser Geräte ist jedoch, dass man sie zum Erfassen von Handgesten auch meist in der Hand halten muss und, neben unterschiedlichen Angewohnheiten solche Geräte zu halten, ist dadurch auch die natürliche Bewegung des Menschen leicht behindert. Hein et al. entschieden sich auch aus diesen Gründen für die Verwendung eines Armbands, welches als „ein autonomes eingebettetes System konstruiert ist“ [HHK09]. Dabei werden die vom Accelerometer erfassten Daten auf dem enthaltenen Mikrocontroller klassifiziert und die erkannte Geste per Bluetooth versendet. Für die Klassifikation der Gesten wurde auf einen Entscheidungsbaum zurückgegriffen. Cho et al. entwickelten für die ersten Mobiltelefone mit Gestenerkennung von Samsung ein System basierend auf Bayesschen Netzen und Support Vector Machines [CCB<sup>+</sup>06]. Choe et al. griffen für ihr Gestenerkennungssystem ebenfalls auf eine Kombination zweier etablierter Algorithmen zurück, nämlich Dynamic Time Warping und k-Means, womit sie den erheblich zunehmenden Berechnungszeiten von Dynamic Time Warping bei langen und vielen Daten entgegen wirkten [CMC10]. Klingmann machte sich in seiner Masterarbeit bei der Entwicklung seiner Gestenerkennungs-App für das iPhone die Eigenschaften von Hidden Markov Modellen zu Nutzen [Kli09]. Eine andere Masterarbeit nahm sich drei Implementationen von Mustererkennungsalgorithmen vor, verglich sie und optimierte den Code für die Verwendung auf einem Smartphone [Nie08]. Dabei stellte sich die Verwendung des Dynamic-Time-Warping-Algorithmus gegenüber Hidden Markov Modellen und künstlichen neuronalen Netzen als zuverlässige und schnelle Methode für die Gestenerkennung heraus. Einen Vergleich verschiedener Algorithmen vollzogen auch Lau und David in [LD10]. Sie untersuchten unter anderem Entscheidungsbäume, Bayessche Netze, die Support Vector Machine und den k-Nearest-Neighbors-Algorithmus auf ihre Eignung zur Bewegungs- beziehungsweise Aktivitätenerkennung. Die Arbeit von Brezmes et al. bezog sich ebenfalls auf die Erkennung von Aktivitäten, wie Gehen, Treppensteigen, etc., wobei k-Nearest-Neighbors zum Einsatz kam [BGC09].

Die meisten der benannten Systeme erreichten, bei einem vorgegebenen Alphabet aus 5 bis 20 Gesten, eine gute Treffsicherheit von 75 bis 98 %. In der Arbeit von Hein et al. war jedoch gut zu beobachten, wie ein System, dass für 5 einfache Gesten sehr gut

funktioniert, bei 11 etwas komplexeren Gesten eine erheblich schlechtere Erkennungsrate aufweist [HHK09]. Für viele der heutigen Anwendungen sind 5 bis 20 Gesten ausreichend, insbesondere unter dem Aspekt, dass viele Gesten zunächst gelernt werden müssen. Benötigt man jedoch eventuell ein größeres Alphabet, um beispielsweise Gebärdensprache zu erkennen, so ist eine mindestens eben so hohe Treffsicherheit bei einem erheblich größerem Alphabet wünschenswert.

## 2.4. Algorithmen

Die soeben aufgeführten Arbeiten nutzen für die Erkennung von Gesten oder Aktivitäten etablierte Verfahren aus der Mustererkennung oder auch Kombinationen dieser. Die im Folgenden beschriebenen Methoden haben sich in diesen Arbeiten für die Anwendung auf mobilen Geräten als geeignet erwiesen.

### 2.4.1. Hidden Markov Modelle (HMMs)

Bei einem Hidden Markov Modell (HMM) handelt es sich zunächst um einen probabilistischen endlichen Automaten dessen Zustandsfolge eine Markovkette bildet. Eine Markovkette hat die Eigenschaft, dass die Wahrscheinlichkeit des Eintretens eines Ereignisses lediglich vom Vorgängerereignis abhängt. Beim HMM sind die eigentlichen Zustände nicht sichtbar, es können lediglich Beobachtungen gemacht werden, die jeweils mit einer bestimmten Wahrscheinlichkeit in einem Zustand auftreten.(Siehe auch [Fin03], [MA07], [Kli09]) Abbildung 2.2 zeigt eine graphische Darstellung eines HMM. Bei der Gestenerkennung wird für jede Geste ein HMM trainiert. Das Training kann dabei beispielsweise durch den sogenannten Baum-Welch-Algorithmus erfolgen. Durch den sogenannten Forward-Algorithmus kann dann festgestellt werden, wie gut ein Modell zu einer Beobachtungsfolge, also Geste, passt und somit das Modell gefunden werden, welches die beobachtete Geste am besten repräsentiert.

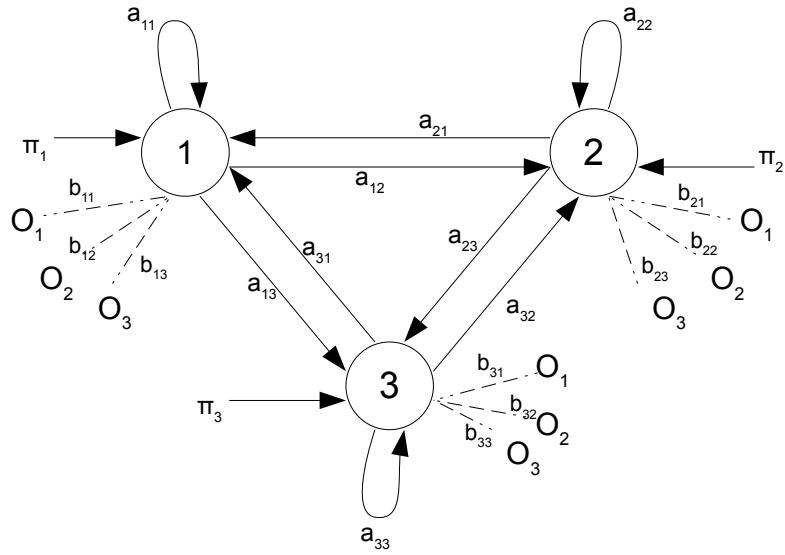
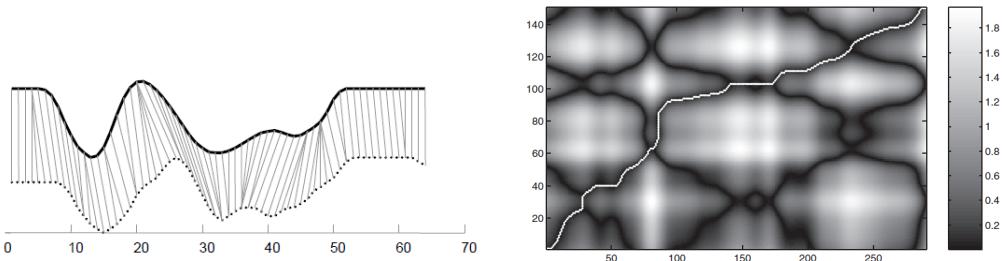


Abbildung 2.2.: HMM mit drei versteckten Zuständen, deren Start- und Übergangswahrscheinlichkeiten ( $\pi_i, a_{ij}$ ), sowie drei Beobachtungen  $O_t$  und deren Auftrittswahrscheinlichkeiten  $b_{it}$

#### 2.4.2. Dynamic Time Warping (DTW)

Beim Dynamic Time Warping ist der Aufwand für das Training minimal, da kein Modell direkt trainiert werden muss, sondern lediglich für jede Geste eine oder mehrere Referenzaufzeichnungen abgespeichert werden. Es wird dann jeweils die beobachte Datensequenz mit einer Referenzsequenz verglichen und die Distanz der beiden ausgerechnet, die beobachtete Sequenz wird dann der Geste zugeordnet, zu deren Referenzaufzeichnungen die geringste Gesamtdistanz besteht. Zum Berechnen der Distanz zweier Sequenzen wird zunächst eine Distanzmatrix aufgestellt, die für jeden Punkt der Sequenz die Distanz (z.B. die euklidische Distanz) zu jedem Punkt der Referenz enthält. Durch diese Matrix wird nun ein Pfad gesucht, der mit minimalen Kosten die Matrix schrittweise durchläuft. Der Pfad bildet sozusagen jeden Punkt der einen Sequenz auf einen der anderen ab. (Abbildung 2.3 bietet eine Visualisierung dieses Prinzips.) Die Kosten dieses Pfades werden als die Distanz der beiden Sequenzen angesehen. Die Suche nach dem Pfad kann durch dynamische Programmierung effizient umgesetzt werden, wobei der eigentliche Pfad dann durch Backtracking erhalten wird. (Siehe auch [Mül07], [KP01])



(a) Abbildung der Punkte einer Sequenz auf eine Referenzsequenz (Quelle: [KP01])  
(b) Pfad durch eine Distanzenmatrix (Quelle: [Mül07])

Abbildung 2.3.: Dynamic Time Warping - Visualisierung

### 2.4.3. Support Vector Machine (SVM)

Eine Support Vector Machine (SVM) bestimmt zur Klassifikation von Daten Trennebenen, wobei nur die sogenannten Stützvektoren (support vectors) einer Klasse, jene die zur möglichen Trennebene am nächsten liegen, ausschlaggebend sind. Bei der Klassifizierung mit zwei Klassen wird die Trennebene so gelegt, dass der Abstand der nächsten Datenpunkte zur Trennebene maximiert wird. Die Mehrheit der Daten sollte außerhalb eines maximierten Bereichs um die Ebene, dem sogenannten Margin, liegen. Um sicherzustellen, dass immer eine Ebene existiert werden in Einzelfällen Ausnahmen zugelassen, die sich wiederum negativ auf den Gesamtabstand auswirken. Es ist möglich bei Nichtlinearität der Daten zunächst eine Ebene in einer höheren Dimension zu bestimmen. Für die Klassifizierung von mehr als zwei Klassen sind nach dem selben Prinzip diverse Multiklassen-Support-Vektor-Maschinen entwickelt worden. (Siehe auch [Bis06])

### 2.4.4. K-Nearest-Neighbors (kNN)

Der K-Nearest-Neighbors-Algorithmus (kNN) verfolgt eine simple Idee zur Klassifikation von Daten. Ähnlich wie beim Dynamic Time Warping ist der Trainingsaufwand minimal, da lediglich für jede Klasse Referenzmuster abgespeichert werden. In der Erkennungsphase werden dann die  $k$  nächsten Nachbarn des zu klassifizierenden Datums ausfindig gemacht, wobei ein geeignetes Abstandsmaß wie zum Beispiel die euklidische Distanz verwendet wird. Das Datum wird anschließend der Klasse zugeordnet zu der die Mehrheit seiner  $k$  Nachbarn gehört. (Siehe auch [MRS09])

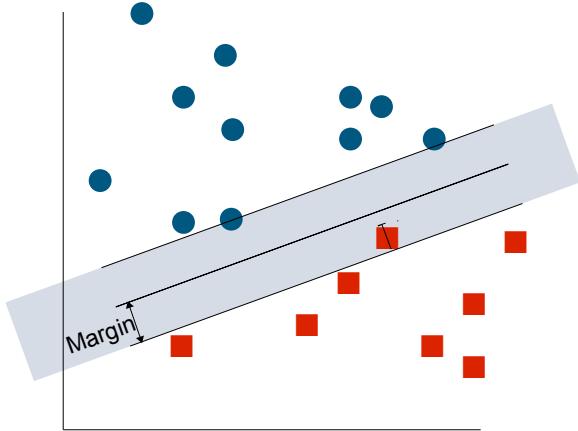


Abbildung 2.4.: eine Soft-Margin SVM

#### 2.4.5. K-Means

Bei K-Means handelt es sich um eine Methode, die sehr häufig im Bereich der Clusteranalyse eingesetzt wird, sich aber auch ohne große Probleme zur Klassifikation verwenden lässt. Für jeden Cluster, bzw. für jede Klasse, wird dabei zunächst ein zufälliger Mittelpunkt ausgewählt, dann wird jeder Datenpunkt der Trainingsmenge, dem nächsten Clusterzentrum zugeordnet. Anhand dieser Zuordnung wird nun ein neuer Mittelpunkt für die bestehenden Cluster berechnet und anschließend folgt wiederum eine neue Zuordnung der Datenpunkte zu den Clusterzentren. Es wird so fortgefahren, bis keine Änderungen mehr eintreten. Möchte man nun ein neues Datum zuordnen, so kann man es einfach der Klasse zuordnen, deren Mittelpunkt dem Datum am Nächsten liegt. (Siehe auch [MRS09], [Bis06])

#### 2.4.6. Entscheidungsbäume

In den Arbeiten [HHK09] und [LD10] wurden unter anderem Entscheidungsbäume zur Klassifikation der Daten eingesetzt. Als Eingabe bei einem solchen Verfahren wird ein Vektor mit den zuvor berechneten benötigten Parametern verwendet, an jedem inneren Knoten des Baumes und der Wurzel wird nun ein Parameter abgefragt und die Blätter, an denen man letztendlich landet, stellen jeweils eine Klasse, also eine bestimmte Geste,

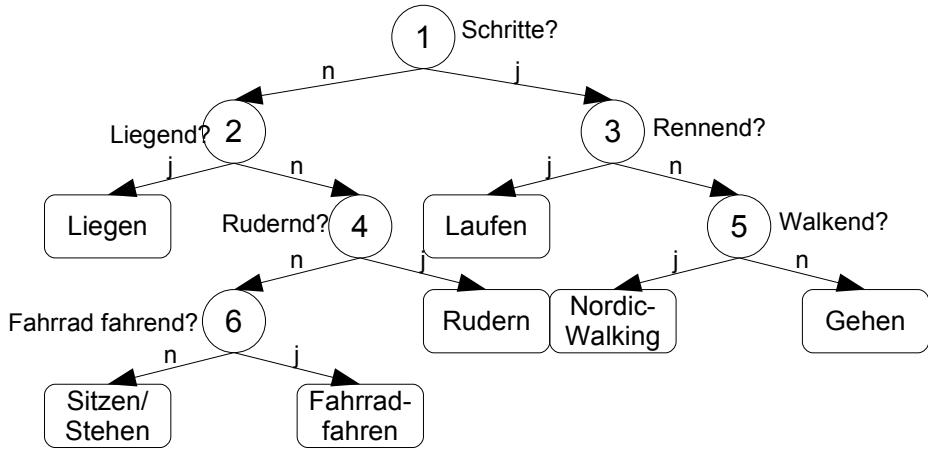


Abbildung 2.5.: Entscheidungsbaum, wie in [PEK<sup>+</sup>06] verwendet

dar. Die Erzeugung eines solchen Baumes kann beispielsweise per Hand, durch persönliche Beobachtungen geschehen, oder automatisch, wie beispielsweise in [HHK09]. Ein Beispiel für einen Entscheidungsbaum in der Aktivitätenerkennung zeigt Abbildung 2.5.

## 2.5. Zusammenfassung

Zusammenfassend lässt sich festhalten, dass für die Gestenerkennung auf mobilen Nutzerendgeräten die selben Herausforderungen wie für andere Benutzerschnittstellen und Gestenerkennungssysteme gelten. Zusätzlich ist jedoch an die verhältnismäßige Ressourcenbeschränktheit von mobilen Geräten zu achten. Im Bezug auf diese Ressourcenbeschränktheit wird die Verwendung von Accelerometern und Gyroskopen gegenüber Bildbasierten Ansätzen häufig bevorzugt. Oftmals wird in verwandten Arbeiten auf Algorithmen der Mustererkennung zurückgegriffen, insbesondere Hidden Markov Modelle, Dynamic Time Warping, die Support Vector Machine, K-Nearest-Neighbors, K-Means und Entscheidungsbäume fanden Anwendung. Nach dieser grundlegenden Betrachtung des Themas werden im folgenden Kapitel die Ergebnisse einer Marktrecherche einen Überblick über die Verwendung von Bewegungsdaten in aktuellen Softwaresystemen bieten.

### 3. Marktrecherche

Die im Rahmen dieser Bachelorarbeit durchgeführte Marktrecherche soll Systeme für mobile Geräte ausfindig machen, die Bewegungsdaten mit Sensoren erfassen und weiterverarbeiten. Es wurde insbesondere nach Apps gesucht, die auf gängigen Plattformen für Smartphones beruhen. Der Schwerpunkt der Suche wurde auf die vielzähligen Anwendungen für Geräte mit Android oder iOS gelegt. Andere Plattformen wie Symbian oder Windows Mobile/ Windows Phone 7 wurden aber auch berücksichtigt. Viele Ideen zur Verwendung von Bewegungsdaten wurden jedoch für mehrere Plattformen implementiert, sich doppelnde Ideen wurden nicht aufgenommen. Die App-Stores der erwähnten Plattformen bieten schon heute eine breite Auswahl an Anwendungen, die die Daten von Beschleunigungs- und immer häufiger auch Neigungssensoren nutzen. Insbesondere konnte eine weite Verbreitung der Eingabe auf Basis von Gesten im Bereich der Spiele beobachtet werden. Sei nun die Steuerung in einem Rennspiel durch das Neigen des Geräts realisiert oder soll durch das Schütteln des Geräts eine virtuelle Dose zum Explodieren gebracht werden. Die Ideen für Anwendungsmöglichkeiten im Entertainment sind vielfältig. Doch gibt es auch immer mehr Apps, die die Bewegungs- oder Lagedaten für den alltäglichen Gebrauch sinnvoll nutzen. So wird ein Smartphone zur Wasserwaage oder bei dem nächsten epileptischen Anfall werden die Angehörigen durch eine SMS automatisch über den momentanen Aufenthaltsort informiert. Die in Tabelle 3.1 dargestellte Auswahl bietet einen Einblick in das Spektrum der gefundenen Apps.

Die recherchierten Applikationen lassen sich in der Art, wie sie mit den Daten des Accelerometers und ggf. des Gyroskops umgehen, unterscheiden. Es gibt Systeme, die:

- die Lage bzw. die Neigung des Geräts erkennen,
- Bewegungen, die eine bestimmte Intensität haben oder in einem bestimmten Frequenzbereich liegen, erkennen,
- komplexe dreidimensionale Gesten aus einem vorgegebenen Alphabet erkennen.

	Name	Betriebssystem	Beschreibung
Neigung	FlipSilent[1]	Android	Drehung des Displays nach unten, schaltet Gerät auf lautlos
	RotoView [2]	iOS	Navigieren (Scrollen der Anzeige) durch Neigung des Geräts
	iHandy Wasserwaage [3]	iOS	Wasserwaage
Schütteln	Fake Me Out Of Here [4]	Android	Auf Schütteln wird ein Anruf simuliert.
	Würfel [5]	Android	Auf Schütteln wird eine Zufallszahl zwischen 1 und 6 generiert
	Soda Shake [6]	Android	Spiel, bei dem durch energiereiches Schütteln eine virtuelle Dose explodieren soll
komplexe Gesten	Don't Touch [7]	Android	Bei Bewegung des Geräts ertönt Alarm
	Step Cal [8]	iOS	Schrittzähler
	EpDetect [9]	Windows Mobile	Bei Erkennung eines epileptischen Anfalls wird SMS mit GPS-Standort und Zeit versandt.
	Super Wand Duel [10]	Android	Spiel, bei dem sich zwei Spieler mit Gesten duellieren
	Move'n Play [11]	iOS	Spiel, bei dem vorgegebene Gesten mit dem Gerät vollführt werden sollen
	Move'n Call [12]	iOS	Zuordnung von Gesten zu Kontakten, Anruf durch die entsprechende Geste
	Hoccer [13]	iOS, Android	One-to-many-Datenübertragung: durch Wurf- und Auffangbewegung
	Gesture Alarm Call [14]	Android	Nummerneingabe durch das Schreiben der Nummern in der Luft
	Motion Gesture Ad [15]	Android	Erkennt Geste und sendet Signal an andere App.

Tabelle 3.1.: Überblick über aktuelle Apps

Eine ähnliche Unterscheidung, in Neigung, Schütteln und Gesten führten auch Cho et al. in [CCB<sup>+</sup>06] auf.

Die Bestimmung des Neigungswinkels des Geräts kann direkt durch ein Gyroskop oder, sollte nur ein Accelerometer zur Verfügung stehen, durch die Verteilung der Gravitationsanteile auf die drei Achsen bestimmt werden. Somit ließe sich beispielsweise, wie heute schon auf annähernd jedem mobilen Gerät integriert, die Bildschirmansicht, an die Lage des Geräts anpassen, das Smartphone ließe sich als Wasserwaage verwenden, oder das Scrollen über Inhalte ließe sich durch Neigung in die gewünschte Richtung realisieren.

Die Erkennung von Schütteln beziehungsweise von Erschütterungen des Geräts umfasst die Analyse, ob das Signal innerhalb eines Zeitintervalls bestimmte Schwellwerte überschreitet [CCB<sup>+</sup>06]. Mit ähnlichem Aufwand lässt sich auch feststellen, ob das Signal in einem bestimmten Frequenzbereich liegt, jedoch ist eine solche Betrachtung in [CCB<sup>+</sup>06] nicht aufgeführt. Es sind Szenarios denkbar, bei denen mit einem Schütteln des Geräts Aktionen, wie das Erzeugen von Zufallszahlen oder das Abspielen eines Alarms, ausgelöst werden, oder aber auch einfache Schrittzähler. Die App EpDetect ist ein Beispiel für eine Anwendung, die auf Bewegungen im Frequenzbereich von 2-5 Hz reagiert, wie sie bei einem epileptischen Anfall auftreten [17].

Für die Erkennung komplexer Gesten im dreidimensionalen Raum werden zumeist die bereits beschriebenen Algorithmen zur Mustererkennung genutzt. In diesem Bereich fallen wiederum viele Spiele auf, aber auch das Wählen von Nummern kann schon heute durch Gesten in der Luft erfolgen.

Für zukünftige Benutzerschnittstellen ist insbesondere letztere Art der Erkennung von Bewegung interessant. Heutzutage ist der Umfang der Gestenalphabete leider noch sehr begrenzt, mit steigender Rechenleistung und Weiterentwicklungen der Algorithmen ist in Zukunft jedoch auch eine zuverlässige Erkennung bei erheblich größeren Alphabeten zu erwarten, womit es möglich wäre Benutzerschnittstellen zu gestalten, die sich der natürlichen Kommunikation des Menschen immer mehr nähern. Es ist auch möglich mit den Verfahren zur Erkennung von dreidimensionalen Bewegungsmustern eine Aktivitätenerkennung zu realisieren, was beispielsweise eine situationsbezogene Assistenz gebrechlicher oder alleinstehender Menschen ermöglichen könnte, wobei lediglich das Telefon an einem festen Platz am Körper befestigt sein müsste. Die denkbaren Einsatzmöglichkeiten für dreidimensionale Gesten sind mannigfaltig. Das Einsatzszenario im anschließenden Kapitel kann als ein weiteres Beispiel betrachtet werden.

### **3.1. Zusammenfassung**

Die im Rahmen dieser Arbeit durchgeführte Marktrecherche zeigt, dass schon heute eine vielfältige Nutzung von Bewegungsdaten auf mobilen Geräten besteht. Einige Systeme reagieren auf die Haltung beziehungsweise die Neigung des Geräts, andere untersuchen die Bewegungsdaten auf ihre Intensität oder ihren Frequenzbereich hin und wieder andere erkennen komplexere Bewegungsmuster im Raum. Letzteres bietet viel Potential für künftige Benutzerschnittstellen und wird auch für das folgende Einsatzszenario verwendet, für welches die Nutzbarkeit einer der recherchierten Softwarelösungen untersucht wird.

## 4. Implementierung

In diesem Teil der Arbeit wird die Auswahl und das Aufsetzen eines der recherchierten Systeme für einen exemplarischen Einsatz beschrieben. Dabei wird zunächst die mögliche Anwendung, daraufhin die Auswahl des benötigten Gestenalphabets und des Systems behandelt. Es folgen eine Darlegung der Schritte zum Aufsetzen des Systems, sowie eine Auswertung der erzielten Erfahrungen.

### 4.1. Beschreibung des Einsatzfeldes

Ein mögliches Einsatzgebiet von Gestenerkennung auf mobilen Geräten liegt in der Unterstützung von Lernprozessen. Hierbei könnte der Einsatz von Bewegung den Lernprozess auf einer zusätzlichen Ebene unterstützen. [PZL11] führt beispielsweise einige Ideen zu diesem Bereich auf. Interessant ist der Einsatz von Bewegung als Eingabe innerhalb von mobilen pervasiven Lernspielen. Diese Spiele haben das Potential durch die Verbindung von Realität und virtuellen Anteilen beim Lernenden positive Stimmungen bzw. Emotionen hervorzurufen, der den Lernstoff dann mit diesen Gefühlen in Verbindung setzen kann, wodurch ein höherer Lernerfolg möglich ist [Ede08]. Wichtig ist bei solchen Spielen insbesondere eine natürliche Interaktion mit den entsprechenden Teilen des Systems, wobei unter anderem die Gestenerkennung hilfreich sein kann.

Zeitgleich zu dieser Arbeit wird am Lehrstuhl für Komplexe Multimediale Anwendungsarchitekturen an der Universität Potsdam im Rahmen einer Diplomarbeit ein pervasives kollaboratives Spiel von Tobias Moebert entwickelt. Es soll den Spielern das Routing in Mobilen Ad-Hoc Netzwerken näher bringen, wobei mögliche Probleme erkannt und in Zusammenhang mit den Anforderungen an ein Ad-Hoc-Routingprotokoll gesetzt werden sollen. Der Spielaufbau setzt sich dabei aus zwei Gruppen zusammen, einer Gruppe an stationären Computern, die versucht über das Ad-Hoc-Netzwerk erfolgreich Nachrichten zu versenden, und einer Gruppe, welche unter Verwendung mobiler Geräte, wie Smartphones, die Knoten des Netzwerks darstellen. Diese Knoten müssen zunächst möglichst lange im Netz bleiben, was sie durch das Einsammeln von virtuellen Batterien erreichen.

Zum Anderen müssen Nachrichten an bestimmte Knoten im Empfangsbereich weitergeleitet werden. Dabei ist das Einleiten von diesen Aktionen der Knoten bisher durch das Drücken von Buttons angedacht. Der Einsatz von Gesten könnte hier eine gute Alternative bieten, da sie das Erleben der Aktion natürlich unterstützen können. Außerdem ist es möglich eine entsprechende Aktion einzuleiten ohne ständig den Bildschirm fixieren zu müssen.

## 4.2. Auswahl der Gesten

Das aufzusetzende Gestenerkennungssystem soll die Einleitung zweier Aktionen durch Gestik unterstützen: das Aufsammeln von Batterien, sowie das Versenden von Nachrichten von einem Knoten zu einem anderen.

Bei der Auswahl der dazugehörigen Gesten ist Diverses zu beachten, wobei der Hauptaugenmerk auf dem Nutzer liegen sollte. Er sollte die Verwendung des Systems als angenehm empfinden. In [Kor08] sind vier Charakteristika für die Auswahl des Gestenalphabets aufgeführt:

- **Intuitiv** Der Nutzer soll das Interface mit geringer oder sogar ohne Einweisung nutzen können.
- **Metaphorisch logisch** Der Nutzer kann die Geste mit der dahinter stehenden Funktionalität verbinden.
- **Einfach zu merken und ohne Zögern ausführbar** Der Nutzer kann sich auf seine Aufgabe und muss sich nicht auf die Interaktion konzentrieren.
- **Ergonomisch** Die Gesten sollten in ihrer Ausführung physisch nicht zu anstrengend sein.

Gesten, die intuitiv und einfach zu merken sind, sind meist eher simpel, sie sollten auch nicht entgegen den Gewohnheiten der Nutzer sein. Nutzt die Mehrzahl bereits existierender Systeme ein gemaltes 'X' zum Beenden eines Programms, so sollte man gut abwägen, ob es sinnvoll ist ein 'O' zu verwenden. Auch eine unnötige Aufblähung des Umfangs des Gestenalphabets ist im Bezug auf das einfache Merken der Gesten eher hinderlich.

Zu beachten bei der Auswahl der Gesten sind immer auch persönliche und kulturelle Unterschiede. Kulturell unterscheiden sich häufig Rhythmus, Frequenz und die Semantik der Gesten [Kor08]. Neben diesen Unterschieden können persönliche Unregelmäßigkeiten in der Haltung des Geräts hinzukommen. Dies könnte lediglich leichte Abweichungen im

Winkel des Geräts betreffen, aber auch Grundlegenderes, wie der Frage in welcher Hand das Gerät bevorzugt gehalten wird.

Neben den Designentscheidungen den Nutzer betreffend sollten auch aus technischer Sicht einige Überlegungen Beachtung finden. So ist es beispielsweise ungünstig, zwei Gesten so zu wählen, dass die eine Teil der anderen ist, da bei einer automatischen Erkennung von Beginn und Ende der Gesten keine eindeutige Zuordnung mehr möglich wäre. Ebenfalls ungünstig sind zwei Gesten deren Bewegungsausführungen sehr ähnlich sind, ein Beispiel hierfür wären die in die Luft gemalten Ziffern '0' und '6'. In [CCB<sup>+</sup>06] stammten allein 30% der gesamten Erkennungsfehler von jenem Paar.

Bei dem beschriebenen Spiel würde ein Spieler sein mobiles Gerät die meiste Zeit bequem in der Hand vor dem Körper halten, sodass der Bildschirm, der ja wichtige Spielinformationen enthält, zu ihm zeigt und gut lesbar ist. Diese Haltung ist im Folgenden als Ausgangsposition für die Gesten zu betrachten.

Für das Aufsammeln von Batterien kamen im Wesentlichen zwei Bewegungsabläufe in Betracht. Sie lehnen sich beide am Aufnehmen von Gegenständen an. Die eine Variante geht von der Vorstellung aus, dass ein Gegenstand etwa in der Größe des Smartphones auf einem Tisch direkt vor einem steht. Um diesen aufzunehmen würde die Hand aus der Ausgangsposition eine Bewegung schräg nach vorne und unten vollführen und anschließend auf ähnlichem Wege leicht versetzt wieder in die Ausgangsposition zurückkehren. Abbildung 4.1(a) zeigt den Verlauf einer möglichen Geste. Die andere Variante könnte man sich vorstellen, als wenn man einen Teddybären in die Arme schließen möchte. Wobei lediglich eine Hand betrachtet wird. Zunächst würden dabei die Arme ausgebreitet und, um den Bären dann zu halten, nach innen wieder geschlossen werden, wobei eine Art Kreisbewegung horizontal zum Boden entsteht (Bild 4.1(b)).

Für das Versenden von Nachrichten kommt die Assoziation zu einem Ballwurf auf. Dies könne in zwei Varianten geschehen. Zum Einen könne man das Smartphone neben den Kopf heben und anschließend in einem Bogen schwungvoll nach vorne bewegen. Zum Anderen kann man einen Ball aber auch von Unten her Werfen, was zu einer schwungvollen Bewegung schräg nach vorne/oben führen würde (Siehe Abbildung 4.2). Eine weitere Alternative wäre hier eine Wurfbewegung wie beim Werfen einer Frisbee-Scheibe.

Bei jeder dieser Bewegungen wurde das Gerät bisher in der rechten Hand des Nutzers gehalten. Für Menschen, die es gewohnt sind mit der linken Hand zu arbeiten, wäre dies eher unangenehm. Eine Aufnahme der spiegelverkehrten Gesten in das System wäre aus diesem Grund wünschenswert und ist auch bei der Auswahl der Gesten zu beachten. Um die grundlegende Einsatzfähigkeit eines bestehenden Systems zu testen, ist jedoch die Hinzunahme dieser zusätzlichen Gesten nicht ausschlaggebend und wurde, um den Auf-



(a) Variante 1: Aufnehmen vom Tisch (b) Variante 2: In den Arm schließen

Abbildung 4.1.: Einsammeln einer Batterie - Gestenauswahl



(a) Variante 1: Wurf neben dem Kopf (b) Variante 2: Wurf von unten

Abbildung 4.2.: Versenden einer Nachricht - Gestenauswahl

wand der Arbeit zu begrenzen, nicht umgesetzt. Aus den erarbeiteten Gesten wurden für das Aufsammeln einer Batterie Variante 1 (Aufheben vom Tisch) und zum Versenden der Nachrichten der Frisbeewurf gewählt. Dies liegt zum Einen an persönlichen Präferenzen zum anderen an bereits angesprochenen Punkten. So ist beispielsweise das Werfen über dem Kopf als Geste zum Versenden der Nachrichten eher weniger geeignet, da dies eine sehr große Bewegung ist, die eventuell nicht von jedem Benutzer als angenehm empfunden würde. Oder wählte man eine Kombination aus Aufheben eines Gegenstandes vom Tisch und dem Werfen eines Balls von unten, könnte bei ungünstiger Haltung des Geräts das Werfen des Balls eine Teilgeste des Aufhebens sein. Die gewählten Gesten hingegen unterscheiden sich genügend und täten dies auch, sollte man die Variante für die linke Hand ins System aufnehmen.

### **4.3. Auswahl des Systems/ der App**

Bei den für das Einsatzszenario ausgewählten Gesten handelt es sich um Gesten im dreidimensionalen Raum. Folglich gilt es ein System auszuwählen, dass diese auch erkennen kann. Zunächst sollten die in Kapitel 3 recherchierten Systeme auf ihre Eignung hin untersucht werden. Es gibt keine Einschränkungen bezüglich des Betriebssystems oder der verwendeten Sensoren. Es ist allerdings erwünscht, jedoch nicht notwendig, dass der Beginn und das Ende der Geste automatisch erfasst werden. Das aufgesetzte System sollte später möglichst ohne persönliches Training auskommen. Trotzdem muss es eine Möglichkeit geben das System um die gewünschten Gesten zu erweitern, ebenfalls ist es unumgänglich, dass die Funktionsweise des Systems, dem Szenario entsprechend angepasst werden kann, also die Möglichkeit die Information über eine erkannte Geste weiter zu verarbeiten. Tabelle 4.1 bietet eine Übersicht über die entsprechenden Eigenschaften der recherchierten Apps. Die Systeme, die über keine automatische Start- und Endpunktterkennung verfügen, lassen den Nutzer, während er eine Geste ausführt, einen Button gedrückt halten. Dies ist zwar nicht allzu komfortabel, aber für den Einsatzzweck ausreichend. Die automatische Erkennung der anderen Systeme beruht meist auf einer relativen Ruheposition. Überschreitet die Geschwindigkeit also einen gewissen Schwellwert wird der Beginn der Geste registriert und bei Unterschreitung des Wertes wird das Gestenende markiert. Ausschlaggebend für die Auswahl des Systems sind, wie bereits erwähnt, die Erweiterbarkeit des Systems um neue Gesten, sowie die mögliche Weiterverarbeitung. Diese Bedingungen werden beide lediglich von einer App erfüllt.

Eine Alternative zur Verwendung einer vorgefertigten App, welche erheblich mehr Anpas-

Name	Super Wand Duel	Move'n Play	Move'n Call	Hoccer	Gesture Alarm Call	Motion Gesture Ad
Entwicklung	Jeremy Vight (Phr00t)	Probayes, Aeon Consulting	Probayes	Hoccer GmbH	le2090	RabiSoft
verwendete Sensoren	Accelerometer	Gyroskop, Accelerometer, Kompass	Gyroskop, Accelerometer, Kompass		Accelerometer	Accelerometer
automatische Start- und Endpunktterkennung	✓	✗	✗		✗	✓
nutzer-unabhängiges Training	✓	✗	✗	✓		✗
Erweiterbarkeit des Alphabets	✗	✗	✓	✗	✗	✓
Weiterverarbeitung	✗	✗	✗	✗	✗	✓

Tabelle 4.1.: Systeme zur Erkennung komplexer Gesten

sungen an die Bedürfnisse des aufzusetzenden Systems erlaubt, wäre die Verwendung von Bibliotheken, die Methoden für die Gestenerkennung bzw. Mustererkennung zur Verfügung stellen. Leider konnten keine kostenfreien Bibliotheken dieser Art ausfindig gemacht werden, die speziell für mobile Geräte konzipiert sind. Das Verwenden anderer Tools wurde vermieden, da Tests bezüglich des Rechenbedarfs und eventuelle Optimierungen für den Einsatz auf mobilen Geräten den Rahmen dieser Arbeit übersteigen.

Insofern wird im Folgenden die App „Motion Gesture Ad“ verwendet werden [15]. Sie bietet die Möglichkeit Referenzmuster selbst aufzunehmen und ordnet folgende Bewegungsmuster diesen Referenzgesten zu. Ist eine Geste erkannt, so wird lediglich eine Nachricht darüber im System verbreitet, diese Nachricht kann mit einer anderen App zur Weiterverarbeitung verwendet werden. Es wäre für spätere Vergleiche durchaus von Interesse, welche Algorithmik in dieser App verwendet wurde, jedoch waren darüber leider keine Informationen zu erhalten.

## 4.4. Aufsetzen des Systems

Voraussetzung für die Verwendung des ausgewählten Systems ist ein Android Gerät (mindestens Android 1.6), welches über ein Accelerometer verfügt. Da kein Smartphone dieser Art für die exemplarische Umsetzung zur Verfügung stand, wurde für einen Testlauf ein Motorola Xoom Tablet verwendet, welches die oben genannten Anforderungen erfüllt, allerdings, aufgrund seiner Größe, nicht komfortabel in der Ausführung der gewählten Gesten ist.

Die Kommunikation mit dem Server, auf welchem ein Spiel läuft, sieht aus Gründen der Flexibilität und Erweiterbarkeit vor, dass sich der Server mit einer IP-Adresse über einen SOAP-Webservice registriert. An die angegebene Adresse werden dann Informationen über aufgetretene Gesten via UDP (User Datagram Protocol) versendet, auf welche dann vom Server reagiert werden kann. Werden die Informationen nicht mehr benötigt, so kann sich der Server wieder über den Webservice abmelden. Dieses Vorgehen hat den Vorteil, dass es möglich ist, dass ohne Probleme mehrere Empfänger Zugriff auf die Informationen über erkannte Gesten erhalten können, nebenbei ist diese Lösung auf Serverseite auch plattformunabhängig. Zunächst wurde daran gedacht einen solchen Webservice direkt auf dem mobilen Gerät bereitzustellen, da jedoch Android keine SOAP-Webservices unterstützt und zu viele Anfragen bei einem ressourcenbeschränktem Gerät bedenklich sind, wurde eine zusätzliche Serveranwendung dazwischen gestellt. Diese Serveranwendung, stellt den beschriebenen Registrierungsservice bereit und versendet an die Spieleserver die Nachrichten, welche sie zuvor vom mobilen Gerät erhalten hat. Damit ein Spieleserver sich nicht für jeden der unter Umständen vielen Mitspieler neu registrieren muss, senden alle mitwirkenden Geräte ihre Gesten an die selbe Anwendung. Dazu wird ein REST(Representational State Transfer)-Webservice angeboten, an welchen die mobilen Geräte die Information über die erkannte Geste sowie ihren Nutzernamen, der im Spiel verwendet wird, senden. Über diesen Nutzernamen kann ein Spieleserver Geste und Spieler einander zuordnen. Zur besseren Übersicht ist die Kommunikationsstruktur in Abbildung 4.3 dargestellt.

Die in Java geschriebene Serveranwendung bildet die Verbindung zwischen den mobilen Geräten der Mitspieler und anderen Teilen des Spiels, die, wie ein möglicher Spieleserver, auf Eingaben des Nutzers durch Gesten warten. Die Schnittstelle zu den mobilen Geräten bildet dabei ein RESTful Webservice, der mit Hilfe des Frameworks Restlet [16] erstellt wurde. Die Registrierung der Empfänger wird über einen SOAP-Webservice angeboten. Dazu wurde mit dem in Java enthaltenen JAX-WS gearbeitet.

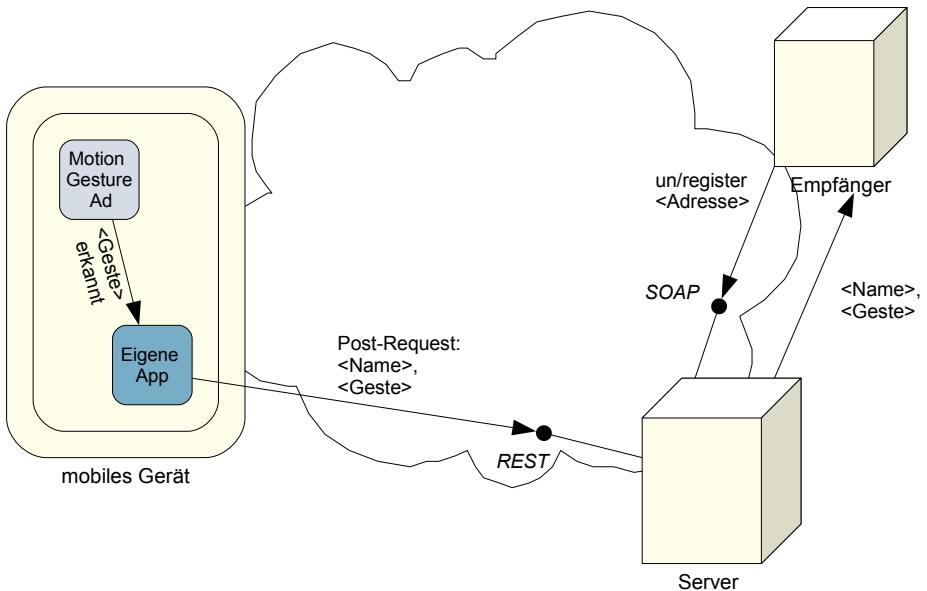


Abbildung 4.3.: Kommunikation im System

Die Android-App beruht lediglich auf den Bibliotheken des Android SDKs, wobei für diese App die Android-Version 3.0 verwendet wurde. Die Anwendung startet nach den Eingaben des Nutzers über die GUI (siehe Abbildung 4.4), welche den eigenen Nutzernamen und die Adresse des RESTful Webservices betreffen, einen Broadcast-Receiver, welcher die Nachrichten (Broadcast Intents) der Motion Gesture Ad, auffängt. Bei Erhalt eines solchen Intents wird ein Android-Service (ein Anwendungsteil, der im Hintergrund operiert und keine Nutzerschnittstelle bietet) in einem separaten Thread gestartet, welcher einen HTTP-Client erzeugt und einen Post-Request an den RESTful Webservice versendet. Damit diese App ihre Aufgabe erfüllen kann ist es zuvor nötig die Motion Gesture Ad zu starten. Der Aufbau der PublishGesture-App ist nochmals in Abbildung 4.5 abgebildet.



Abbildung 4.4.: Interface der Android-App zur Dateneingabe

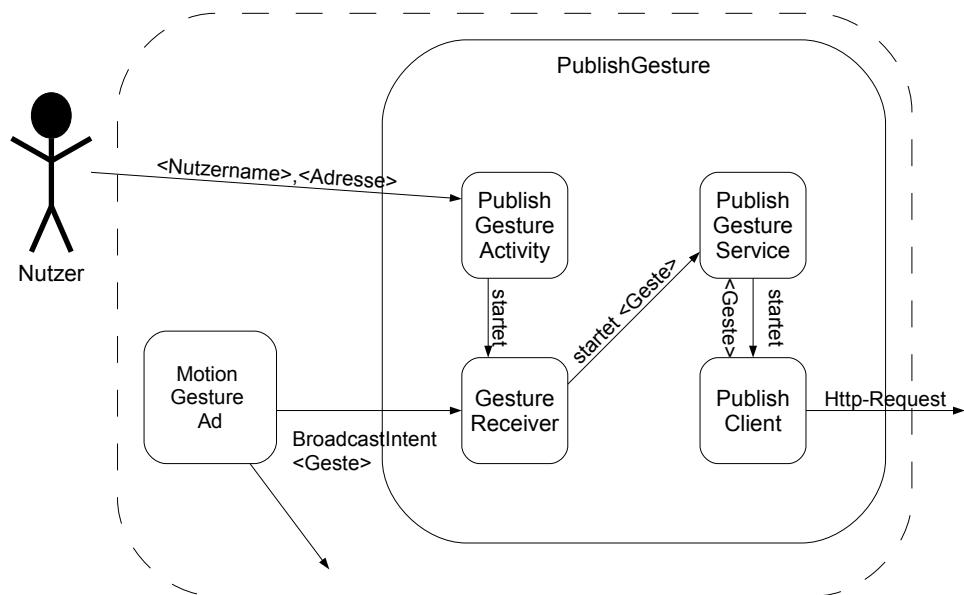


Abbildung 4.5.: Struktur der Android-App

Neben der Implementierung der Programmteile, die der Weiterverarbeitung der erkannten Geste dienen, ist es nötig die entsprechenden Referenzmuster aufzuzeichnen. Im Grunde bietet die App Motion Gesture Ad die Möglichkeit ein Referenzmuster je Geste aufzuzeichnen (Abbildung 4.6). Da dies aber häufig zu falschen Einordnungen der Gesten führte, ist es sinnvoll mehrere Referenzmuster pro Geste in den Algorithmus einfliessen zu lassen. Die Notwendigkeit mehrerer Referenzmuster wird insbesondere dadurch verschärft, dass sich die Mitspieler frei im Raum bewegen sollen, was zusätzliche Beschleunigungen in die Bewegungsrichtung zur Folge hat und auch nicht immer garantiert werden kann, dass die Spieler vor und nach der Gestenausführung still stehen. Letzteres könnte problematisch werden, da das System den Start- und Endpunkt einer Geste anhand der relativen Ruheposition des Geräts ausmacht. Eine derartige Verwendung mehrerer Referenzen ist allerdings in dem verwendeten System so nicht vorgesehen. Um dennoch die fehlerhafte Erkennung einzuschränken wurde beim Training der Gesten, jedes falsch oder nicht eingeordnete Muster als neue Geste in die Gestenliste aufgenommen, sodass es mehrere Gesten gibt, die die selbe Nachricht erzeugen. Also werden zu unserer Wurfbewegung mehrere eigenständige Bewegungsmuster abgespeichert, die alle das Throw-Intent erzeugen.



Abbildung 4.6.: Erweiterung der App 'Motion Gesture Ad'

## 4.5. Evaluation

Der Test des entstandenen Systems, umfasst nicht nur die Prüfung der Funktionalität der Klassen, sondern auch die Schätzung des Fehlers bei der Erkennung der Gesten. Häufig wird für diesen Zweck eine große Datensammlung der ausgeführten Gesten erhoben, welche sowohl Trainings- als auch Testdaten umfasst. Mit statistischen Methoden, wie der N-Fold-Cross-Validation wird dann der Fehler geschätzt. Solche Methoden sind in diesem Fall leider nicht möglich, da das ausgewählte System 'Motion Gesture Ad' weder Möglichkeiten zum Import noch zum Export von Referenzaufzeichnungen bietet. Dementsprechend wurde nach dem Aufnehmen von Referenzmustern ein Live-Test mit 6 Probanden durchgeführt. Diese wiederholten in unterschiedlicher Reihenfolge immer wieder die zwei ausgewählten Gesten, wobei sie eine Folge im Stand ausführten und eine weitere, während sie sich durch den Raum bewegten. Die Ergebnisse dieses Testlaufs waren leider nicht ausreichend (Siehe Tabelle 4.2). Die gesamte Erkennungsrate von 60,7% ist sehr

	Pick	Throw	insgesamt
im Stand	58,0%	56,3%	57,1%
in Bewegung	70,9%	58,3%	64,4%
insgesamt	64,4%	57,3%	60,7%

Tabelle 4.2.: Erkennungsrate des Systems

niedrig und wurde von den Probanden als störend empfunden. Zumal in diese Rechnung noch nicht eingeflossen ist, dass während der Bewegung im Raum immer wieder Gesten erkannt worden sind, obwohl keine beabsichtigte Gestenausführung statt fand.

Als Interface für das beschriebene Szenario ist das System aufgrund seiner schlechten Erkennungsrate eher nicht geeignet. Es könnten unter Umständen bessere Ergebnisse mit anderen Referenzgesten erzielt werden, jedoch wurde ein weiterer Versuch mit neuen Referenzgesten noch nicht unternommen, da durch die fehlenden Import- und Exportmöglichkeiten, bei schlechteren Ergebnissen, der vorherige Zustand nicht wiederhergestellt werden könnte. Zudem war aus zeitlichen Gründen ein weiteres Treffen mit den Probanden nicht möglich.

## 4.6. Zusammenfassung

Für den Einsatz im beschriebenen pervasiven Lernspiel eignete sich, aufgrund von fehlenden Anpassungsmöglichkeiten, nur eines der recherchierten Systeme. Das Aufsetzen

des Systems umfasste die Auswahl geeigneter Gesten für zwei Aktionen, sowie die Anpassung des Systems an die Anforderungen des Einsatzszenarios. Leider erwies sich die Verwendung des ausgewählten Systems als umständlich, da Referenzgesten, weder importiert noch exportiert werden können und lediglich eine Erkennungsrate von 60,7% erreicht werden konnte.

## 5. Fazit und Ausblick

Wenn man aktuelle Systeme zur Gestenerkennung auf mobilen Geräten untersucht, so kann man feststellen, dass die Gestenerkennung auf mobilen Nutzerendgeräten bereits in vielen Einsatzbereichen weit über die Unterhaltung hinaus Einzug erhalten hat. So existieren bereits Apps, die den Nutzer in alltäglichen Situationen unterstützen können (z.B. die Wasserwaage oder das Wählen von Nummern durch Gesten). Auch im medizinischen Bereich gibt es Entwicklungen, so sollen die Aktivitäten der Nutzer überwacht werden, um Ausnahmesituationen festzustellen und darauf reagieren zu können. Ein simples Beispiel dieser Art ist die Erkennung von epileptischen Anfällen durch die App EpDetect [9].

Die vorgestellten Arbeiten und Apps, sowie das in der Arbeit realisierte Einsatzbeispiel zeigen, dass, bei kleinen Gestenalphabeten, einer geschickten Gestenauswahl und einem ausführlichen Trainingsprozess der Algorithmik, relativ zuverlässige Systeme möglich sind. Jedoch reichen die heutigen Leistungen noch nicht für komplexere Anwendungen, wie beispielsweise der Erkennung von Gebärdensprachen, auch sind noch keinesfalls Anwendungen möglich, bei denen eine Erkennung der Gesten von 100% notwendig sind, da unter Umständen Menschenleben davon abhängen könnten.

Gedenkt man selbst ein System für einen speziellen Einsatz zu erstellen, ist von der Nachnutzung bereits existierender Apps bisher eher abzuraten. Die meisten Systeme bieten keine oder nur wenige Möglichkeiten zur Anpassung an die Bedürfnisse des Einsatzszenarios, zudem sind Informationen über die verwendete Algorithmik nur selten verfügbar. Das für diese Arbeit gewählte System ist zwar um neue Gesten erweiterbar und bietet Möglichkeiten zu deren individueller Weiterverarbeitung, der effektive Einsatz scheitert jedoch an einer schlechten Erkennungsrate und der fehlenden Möglichkeit zum Import und Export von Referenzmustern, was zum Einen ein nachvollziehbares Testen erschwert, zum Anderen die Installation des trainierten Systems auf anderen Geräten verhindert.

Für den eigenen Einsatz bietet es sich eher an ein Gestenerkennungssystem von Grund auf neu aufzusetzen, unter Umständen unter Verwendung entsprechender Frameworks. Leider konnten im Rahmen der Recherche zu dieser Arbeit keine kostenfreien Biblio-

theken speziell für mobile Geräte ausfindig gemacht werden. Die Nutzung allgemeiner Toolkits aus dem Bereich Mustererkennung ist auch möglich, setzt womöglich aber entsprechende Optimierungen voraus.

Für eventuell anschließende Arbeiten wäre eine solche Implementierung eines Gestenerkennungssystems interessant. Es wäre sicherlich ebenfalls von Interesse die Möglichkeit einer plattformunabhängige Lösung (z.B. über JavaScript) zu überdenken.

# Literaturverzeichnis

- [BGC09] BREZMES, Tomas ; GORRICO, Juan-Luis ; COTRINA, Josep: Activity Recognition from Accelerometer Data on a Mobile Phone. In: *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*. Berlin, Heidelberg : Springer-Verlag, 2009 (IWANN '09). – ISBN 978-3-642-02480-1, 796–799
- [Bis06] BISHOP, Christopher M.: *Pattern Recognition and Machine Learning*. Springer, 2006
- [CCB<sup>+</sup>06] CHO, Sung-Jung ; CHOI, Eunseok ; BANGA, Won-Chul ; YANG, Jing ; SOHN, Junil ; KIM, Dong Y. ; LEE, Young-Bum ; KIM, Sangryong: Two-stage Recognition of Raw Acceleration Signals for 3D-Gesture-Understanding Cell Phones. In: *10th International Workshop on Frontiers in Handwriting Recognition*, 2006
- [CMC10] CHOE, BongWhan ; MIN, Jun-Ki ; CHO, Sung-Bae: Online Gesture Recognition for User Interface on Accelerometer Built-in Mobile Phones. In: WONG, Kok (Hrsg.) ; MENDIS, B. (Hrsg.) ; BOUZERDOUM, Abdesselam (Hrsg.): *Neural Information Processing. Models and Applications* Bd. 6444. Springer Berlin / Heidelberg, 2010, S. 650–657
- [Ede08] EDEGGER, Francika: *Pervasive Gaming als ein neuer Weg zur Beeinflussung von Denken und Handeln: Eine Anwendung im Lernkontext*. Gabler, 2008 (Gabler Edition Wissenschaft)
- [Fin03] FINK, G.A.: *Mustererkennung mit Markov-modellen: Theorie-praxis-anwendungsgebiete*. Teubner, 2003 (Leitfäden und Monographien der Informatik)
- [GBB10] GUSTAFSON, Sean ; BIERWIRTH, Daniel ; BAUDISCH, Patrick: Imaginary Interfaces: Spatial Interaction with Empty Hands and without Visual Feedback.

- In: *UIST '10 Proceedings of the 23nd annual ACM symposium on User interface software and technology*. New York, USA : ACM, Oktober 2010 (UIST '10), 3-12
- [HHK09] HEIN, Albert ; HOFFMEYER, André ; KIRSTE, Thomas: Utilizing an Accelerometric Bracelet for Ubiquitous Gesture-Based Interaction. In: STEPHANIDIS, Constantine (Hrsg.): *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part II*. San Diego, CA, USA : Springer, 2009 (Lecture Notes in Computer Science), S. 519–527
- [Kli09] KLINGMANN, Marco: *Accelerometer-Based Gesture Recognition with the iPhone*. London, Goldsmiths University of London, Masterarbeit, September 2009
- [Kor08] Kapitel 3 Gesture Interfaces. In: KORTUM, Philip: *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces*. Elsevier/Morgan Kaufmann, 2008 (Morgan Kaufmann series in interactive technologies), S. 75 – 106
- [KP01] KEOGH, Eamonn J. ; PAZZANI, Michael J.: Derivative Dynamic Time Warping. In: *In First SIAM International Conference on Data Mining (SDM'2001*, 2001
- [LD10] LAU, Sian L. ; DAVID, Klaus: Movement recognition using the accelerometer in smartphones. In: CUNNINGHAM, Paul (Hrsg.) ; CUNNINGHAM, Miriam (Hrsg.): *Future Network and MobileSummit 2010*. Florence, Italy, June 16-18 2010
- [MA07] MITRA, Sushmita ; ACHARYA, Tinku: Gesture Recognition: A Survey. In: *IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews* Bd. 37, 2007
- [Mül07] Kapitel 4. Dynamic Time Warping. In: MÜLLER, Meinhard: *Information Retrieval for Music and Motion*. Springer, 2007, 69-84
- [MRS09] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *An Introduction to Information Retrieval*. Cambridge University Press, 2009  
<http://www.informationretrieval.org/>

- [Nie08] NIEZEN, Gerrit: *The optimization of gesture recognition techniques for resource-constrained devices*, University of Pretoria, South Africa, Masterarbeit, 2008
- [PEK<sup>+</sup>06] PÄRKKÄ, Juha ; ERMES, Miikka ; KORPIPÄÄ, Panu ; MÄNTYJÄRVI, Jani ; PELTOLA, Johannes ; KORHONEN, Ilkka: Activity Classification Using Realistic Data From Wearable Sensors. In: *IEEE Transactions on Information Technology In Biomedicine* Bd. 10, 2006
- [PZL11] PFEIFFER, Linda ; ZENDER, Raphael ; LUCKE, Ulrike: Gestenerkennung auf mobilen Geräten: Aktueller Stand und Potential für das Lernen, 2011

# Internetquellen

- [1] <https://market.android.com/details?id=com.maweilabs> (Letzter Zugriff: 5.8.2011)
- [2] <http://www.rotoview.com/> (Letzter Zugriff: 5.8.2011)
- [3] <http://itunes.apple.com/de/app/ihandy-wasserwaage/id299852753?mt=8> (Letzter Zugriff: 5.8.2011)
- [4] <https://market.android.com/details?id=com.incrediapp.fake.call.shake.me.out.of.here> (Letzter Zugriff: 5.8.2011)
- [5] <https://market.android.com/details?id=com.frankmeyeredv.virtualdie> (Letzter Zugriff: 5.8.2011)
- [6] <https://market.android.com/details?id=com.anbeans.SodaShake> (Letzter Zugriff: 5.8.2011)
- [7] <https://market.android.com/details?id=net.mnocompany.beachalarm> (Letzter Zugriff: 5.8.2011)
- [8] <http://itunes.apple.com/de/app/step-cal-der-schrittzhler/id291237868?mt=8> (Letzter Zugriff: 5.8.2011)
- [9] <http://www.epdetect.com> (Letzter Zugriff: 5.8.2011)
- [10] <https://market.android.com/details?id=com.wandduel> (Letzter Zugriff: 5.8.2011)
- [11] <http://www.my-smart-phone.com/move-n-play> (Letzter Zugriff: 5.8.2011)
- [12] <http://www.my-smart-phone.com/move-n-call> (Letzter Zugriff: 5.8.2011)
- [13] <http://hoccer.com/> (Letzter Zugriff: 5.8.2011)
- [14] <https://market.android.com/details?id=kbk.android.gesture.gesturealarmcall> (Letzter Zugriff: 5.8.2011)

- [15] <https://market.android.com/details?id=RabiSoft.MotionGestureAd> (Letzter Zugriff: 5.8.2011)
- [16] <http://www.restlet.org/> (Letzter Zugriff: 10.8.2011)
- [17] *EPDETECT MOBILE PHONE APPLICATION* automatically detects epileptic seizures. – [http://www.epdetect.com/EpDetect\\_user\\_manual.pdf](http://www.epdetect.com/EpDetect_user_manual.pdf) (letzter Zugriff: 5.8.2011)
- [18] *Technische Daten des iPhone 4.* – <http://www.apple.com/de/iphone/specs.html> (Letzter Zugriff: 8.8.2011)

## A. Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

## B. Source Code

Der zugehörige Code zu der Serveranwendung und der eigenen Android App ist auf der beiliegenden CD zu finden.