Universität Koblenz-Landau
Campus Koblenz
Fachbereich 4
Institute for Web Science and Technologies

Bachelorarbeit

„Bewegungserkennung mit Smartphones mittels deren Sensoren“

von

Sven Milker
B.Sc. Informationsmanagement
6. Semester
Matrikel-Nummer: 209110013

Betreuer:

Sebastian Magnus
Prof. Dr. Steffen Staab

Erstprüfer:

Prof. Dr. Steffen Staab

Zweitprüfer:

Sebastian Magnus

Erklärung nach §23 Abs. 7 Satz 3 PO:

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Arbeit von mir selbstständig, ohne fremde Hilfe angefertigt worden ist. Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere, außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenden Abbildungen, Skizzen und Tabellen. Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Die eingereichte schriftliche Fassung entspricht der auf dem elektronischen Speichermedium (CD-Rom).

Waldesch, den 20.07.2012

Sven Milker

Zusammenfassung

Die Erkennung von Bewegungen mit einem Smartphone ist durch die internen Sensoren, mit denen es ausgestattet ist, ohne externe Sensoren möglich. Zunächst werden vergangene Untersuchungen und deren Verfahrensweise betrachtet. Von diesen Untersuchungen wird die hier verwendete Umsetzung der Bewegungserkennung abgeleitet und beschrieben. Ein Großteil der Literatur verwendet ausschließlich den Beschleunigungssensor für die Bewegungserkennung. Diese Bachelorarbeit untersucht dazu den Nutzen weiterer Sensoren, wie z.B. das magnetische Feld, die lineare Beschleunigung oder das Gyroskop. Die Bewegungserkennung erfolgt durch maschinelles Lernen mit Klassifizierungsalgorithmen. Es werden Decision Tree, Naive Bayes und Support Vector Machines verwendet. Zunächst werden die benötigten Sensordaten mit Hilfe einer eigens entwickelten Applikation von Testpersonen gesammelt und gespeichert. Diese Daten werden als Trainingsdaten für die Klassifizierungsalgorithmen benötigt. Das Ergebnis ist ein Modell, das die Struktur der gelernten Daten enthält. Validiert werden die Modelle mit Testdatensätzen, die sich von den Trainingsdatensätzen unterscheiden.

Die Ergebnisse bestätigen vergangene Untersuchungen, dass die Daten des Beschleunigungssensors ausreichend sind, um Bewegungen sehr genau erkennen zu können. Orientierung, Gyroskop und die lineare Beschleunigung hingegen sind nur bedingt für die Bewegungserkennung einsetzbar. Außerdem scheint für eine geringe Anzahl an Testdaten der Decision Tree am besten geeignet zu sein, wenn Bewegungen von Benutzern erkannt werden sollen, von denen keine Trainingsdaten bei der Erstellung des Modells vorhanden sind.

Abstract

Activity recognition with smartphones is possible by using its internal sensors without using any external sensor. First of all, previous works and their techniques will be regarded and from these works an own implementation for the activity recognition will be derived. Most of the previous works only use the accelerometer for the activity recognition task. For this reason, this bachelor thesis analyzes the benefit of further sensors, such as the magnetic field, the linear acceleration or the gyroscope. The activity recognition is performed by classification algorithms. Decision Tree, Naive Bayes and Support Vector machines will be used. Sensor data of subjects will be collected and saved by using an own developed application. This data is needed as training data for the classification algorithms. The result is a model which represents the structure of the data. To validate the model, a test dataset will be used which is different from the training dataset.

The results confirm previous works which indicated that the activity recognition task is possible by only using the accelerometer. Orientation, gyroscope and linear acceleration cannot be used for all problems of the activity recognition. Apart from that, the Decision Tree seems to be the best classification algorithm if the model has no training data of the current user.

Inhaltsverzeichnis

1 Einleitung

Die Erkennung alltäglicher Bewegungen einer Person mit Hilfe von Smartphones ist Gegenstand jüngerer Untersuchungen und in verschiedenen Einsatzgebieten relevant. Diese erstrecken sich vom Gesundheitswesen [1] über soziale Netzwerke [2] bis zum Echtzeit-Tracking öffentlicher Verkehrsmittel [3]. Eine Auflistung dieser Einsatzgebiete ist in [4] zu finden. Alltägliche Bewegungen sind z.B. Gehen, Laufen oder Fahrradfahren, aber auch Stehen oder Sitzen gehören dazu. Jede Bewegung kann in einem Kontext betrachtet werden. Dieser kann entweder die Bewegung sein, die vor oder nach dem betrachteten Zeitraum stattgefunden hat, oder Ort und Zeit der Bewegung, was z.B. für soziale Netzwerke interessant sein kann. Diese Arbeit untersucht die Bewegung der Person ohne jeglichen Kontext. Im Folgenden wird der Begriff *Bewegungserkennung* verwendet, der die Erkennung von Gehen, Laufen, Sitzen etc. einer Person bezeichnet.

Die Bewegungserkennung basiert auf den Messwerten der Sensoren des Smartphones. Ein Sensor ist ein technisches Bauteil, das Signale oder Impulse messen kann [5]. Frühere Untersuchungen verwendeten spezielle Sensoren, die an verschiedenen Positionen des Körpers befestigt wurden. So wurde z.B. ein Beschleunigungssensor am Handgelenk [6], im Schuh [7] oder am Gürtel [8] befestigt. Da es mittlerweile Smartphones gibt, die mit solchen Sensoren ausgestattet sind, kann die Bewegungserkennung ohne diese zusätzliche Hardware durchgeführt werden. Die Smartphones enthalten dabei nicht nur den Beschleunigungssensor, mit dem die meisten Experimente durchgeführt worden sind, sondern auch andere Sensoren, die für die Bewegungserkennung nützlich sind. Dazu gehören z.B. das Gyroskop oder das magnetische Feld (Kompass).

Die Verwendung der Sensoren eines Smartphones erschwert die Bewegungserkennung gegenüber externen Sensoren, da die Position und Ausrichtung des Smartphones unbekannt ist. So könnte sich das Smartphone beim Gehen in der Brusttasche weniger bewegen als in der vorderen Hosentasche. Das bedeutet auch, dass die Sensordaten des Beschleunigungssensors sich signifikant ändern [3, p. 3]. Dies gilt auch für die weiteren Sensoren des Smartphones, auf die genauer in Kapitel 2.1 eingegangen wird, um deren Verwendung für die Bewegungserkennung zu untersuchen.

Diese Arbeit betrachtet vergangene Untersuchungen und schafft einen Überblick über die theoretische Umsetzung der Bewegungserkennung. Auf Basis dieser Untersuchungen wird die konzeptionelle Umsetzung beschrieben, die konkretere Informationen zur eigenen Umsetzung der Bewegungserkennung enthält. Um im Voraus einen Überblick zu verschaffen, wird die notwendige Vorgehensweise kurz beschrieben. Die Details behandelt Kapitel 3.

Um Bewegungen erkennen zu können, müssen die charakteristischen Eigenschaften jeder Bewegung bekannt sein. So ist z.B. Laufen im Gegensatz zu Stehen eine dynamischere Bewegung, was sich in den Sensordaten widerspiegelt. Um die genauen Eigenschaften herauszufinden, werden Sensordaten der einzelnen Bewegungen benötigt, die mit einer eigenen Android-Applikation gesammelt werden. Diese Sensordaten werden in kurze Abschnitte aufgeteilt und für jeden Abschnitt statistische Werte bestimmt, wie z.B. die Standardabweichung für den Beschleunigungssensor. Diese berechneten Werte werden dann mit der angegebenen Bewegung für die Trainingsphase eines Klassifizierungsalgorithmus verwendet. Das Ergebnis der Trainingsphase ist ein Modell, das die Struktur der verwendeten Trainingsdaten enthält. Die finale Android-Applikation verwendet dieses Modell und versucht, die momentane Bewegung des Benutzers zu erkennen und auf dem Display

auszugeben. Es sind verschiedene Szenarien denkbar und die direkte Auswertung und Ausgabe nur eine Möglichkeit, die Bewegungserkennung zu verwenden.

1.1 Motivation

Eines dieser Szenarien gehört z.B. in den bereits erwähnten Bereich des Gesundheitswesens. So kann eine Aufzeichnung der Bewegungen, die ein Benutzer an einem Tag durchgeführt hat, einem Arzt zu einer besseren Diagnose helfen [9]. In [1] wurde über die erkannte Bewegung der Kalorienverbrauch gemessen und ein Tagesdiagramm erstellt. In beiden Fällen ist keine Live-Auswertung der Daten notwendig. Die Daten könnten gesammelt und später ausgewertet werden.

Diese Arbeit behandelt die direkte Live-Auswertung der Sensordaten, für die es ebenfalls denkbare Szenarien gibt. So kann ein Navigationsgerät auf den Fußgängermodus schalten, wenn die Bewegung des Benutzers von *Autofahren* auf *Gehen* wechselt [4, p. 5]. Der Fußgängermodus könnte den Ausschnitt der Karte vergrößern, mehr Details anzeigen und überflüssige Informationen ausblenden, wie z.B. die erlaubte Höchstgeschwindigkeit.

Das Nike + iPod System [7] wertet Sensordaten eines Beschleunigungssensors im Schuh aus und stellt Informationen über Dauer, Laufstrecke, Geschwindigkeit und Kalorienverbrauch zur Verfügung. Dazu kann die Songauswahl an das Laufen angepasst werden. In diesem Fall wird eine Bewegungserkennung benötigt, die Sensordaten direkt auswertet und verarbeitet.

Ebenfalls kann es zur Sicherheit des Benutzers beitragen, wenn das Smartphone auf bestimmte Situationen reagiert. Beim Fahrrad- oder Autofahren sollte man das Smartphone nicht benutzen und keine Anrufe entgegennehmen. Wenn das Smartphone die Bewegung *Fahrradfahren* erkennt, kann es sich auf lautlos schalten oder Anrufe direkt auf die Mailbox umleiten [10].

1.2 Aufgabenstellung

Diese Arbeit untersucht verschiedene Aspekte der Bewegungserkennung mit Smartphones. So werden z.B. die Sensoren des Smartphones verglichen, die aufgrund ihrer unterschiedlichen Funktionen verschiedene Daten sammeln und somit womöglich mehr oder weniger relevant für die Bewegungserkennung sind. Außerdem werden Auswirkungen äußerer Einflüsse untersucht. Dazu gehört z.B. das Trageverhalten des Benutzers, bei dem die Positionierung des Smartphones eine wichtige Rolle spielt. Der Aufenthaltsort des Benutzers kann für die Verfügbarkeit von Sensoren relevant sein, da z.B. das Global Positioning System (GPS) in einer U-Bahn möglicherweise nicht verfügbar ist. Dies überschneidet sich mit technischen Restriktionen, die ebenfalls untersucht werden. Für die Erkennung von Bewegungen werden die drei Klassifizierungsalgorithmen Decision Tree, Naive Bayes und Support Vector Machines verwendet und verglichen.

Für die Erkennung von Bewegungen sind Trainingsdaten notwendig, die entweder selber aufgenommen oder von Dritten bezogen werden müssen. Da die vorhandenen Applikationen für eine Datensammlung nicht den Anforderungen entsprechen und keine geeigneten Daten von Dritten gefunden werden konnten, ist eine eigene Applikation für diese Aufgabe geschrieben worden. Diese wird in Kapitel 4.1.3 genauer beschrieben.

Auf Basis der Ergebnisse wird eine Android-Applikation erstellt, welche für die Erkennung der Bewegung zuständig ist. Die Bewegungserkennung erfolgt über die Messwerte der Sensoren, die ausgewertet und verarbeitet werden. Die Applikation muss dabei die folgenden Anforderungen erfüllen:

- Erfassen von Sensordaten mit einem Smartphone
- Ausschließliche Verwendung von Sensoren des Smartphones
- Erkennung verschiedener Bewegungen durch maschinelles Lernen
- Darstellung der erkannten Bewegung auf dem Smartphone
- Die Erkennungsrate der Bewegungen muss mindestens 90 % sein
- Der Stromverbrauch muss in einem akzeptablen Bereich liegen
- Die Applikation darf keine hohe Auslastung des Smartphones hervorrufen
- Die Bewegungserkennung darf nicht von der Ausrichtung des Smartphones abhängen

1.3 Related Works

Die ersten Untersuchungen der Bewegungserkennung verwendeten einen oder mehrere externe Sensoren, die an bestimmten Positionen des Körpers fest positioniert waren. In [1] wurden drei Beschleunigungssensoren verwendet, die fest an Oberschenkel, Taille und Hand befestigt wurden. Dadurch konnten neben Bewegungen wie Sitzen, Gehen oder Laufen auch Händeschütteln, Zähneputzen oder Staubsaugen bestimmt werden. Dies war nur durch den Beschleunigungssensor am Handgelenk möglich, der die Bewegungen der Hand bzw. des Arms aufgenommen hat.

Dieser Ansatz ist nicht sehr benutzerfreundlich, da die externen Beschleunigungssensoren extra angebracht werden müssen und der Beschleunigungssensor der Hand an einem Handschuh befestigt ist, der im Alltag hinderlich ist. Durch die Verwendung eines Smartphones als Sensor ergeben sich verschiedene Vor- und Nachteile.

Vorteile

- Es werden keine externen Sensoren benötigt, die vor der Benutzung angebracht werden müssen
- Smartphones sind Gegenstände des Alltags geworden, die fast immer mitgenommen werden
- Die Bewegungserkennung kann als Applikation auf dem Smartphone laufen
- Es können alle internen Sensoren des Smartphones verwendet werden

Nachteile

- Die Position und Ausrichtung des Smartphones ist unbekannt und die Messwerte bei gleicher Bewegung möglicherweise unterschiedlich
- Das Smartphone kann locker oder fest in der Tasche liegen, was zu unterschiedlichen Messwerten führen kann
- Es können nur Sensordaten von einer Position gleichzeitig gemessen werden
- Eine Benutzerinteraktion kann die Messungen der Sensoren beeinflussen und somit eine verfälschte Erkennung der Bewegung hervorrufen, wie z.B. beim Schreiben einer SMS während dem Gehen

Tabelle 1 Artikelübersicht

#	Zusätzliche Sensoren	Untersuchte Bewegungen	Anzahl der Positionen des Smartphones	Klassifizierungsalgorithmen
[3]	GPS, WiFi	Gehen, nicht-Gehen, in Fahrzeug	7	Decision Tree
[11]	-	Sitzen, Gehen, Laufen, Fahrradfahren	1	Naive Bayes
[12]	-	Ruhen(Sitzen), Gehen, Treppen, Laufen	5	Andere
[13]	-	Sitzen, Stehen, Liegen, Gehen, Treppen, Fahrradfahren	3	Andere
[14]	GPS, Mikrofon, WiFi	Ruhen, Gehen, Laufen, in Fahrzeug	Beliebig	Decision Tree
[15]	-	Sitzen, Stehen, Liegen, Gehen, Laufen, Springen	3	Andere
[16]	-	Ruhen, Gehen, Laufen, Fahrradfahren, Treppen, in Fahrzeug	6	Support Vector Machine
[17]	-	Gehen, Treppen, Fallen	Beliebig	Andere
[18]	-	Ruhen, Gehen, Laufen, Fahrradfahren, in Fahrzeug	Beliebig	Decision Tree, Naive Bayes, Support Vector Machine, Andere
[9]	-	Stehen, Gehen, Laufen, in Zug	Beliebig	Support Vector Machine, Andere
[19]	Kameraintensität, Nähe	Sitzen, Stehen, Gehen, Laufen	7	Naive Bayes, Andere
[20]	-	Liegen, Sitzen, Stehen, Gehen, Treppen, Laufen[1]	Beliebig	Andere
[21]	GPS	Ruhen, Gehen, Laufen, Fahrradfahren, in Fahrzeug	Beliebig	Decision Tree, Andere
[2]	Mikrofon, GPS	Sitzen, Stehen, Gehen, Laufen	3	Decision Tree
[22]	-	Sitzen, Stehen, Gehen, Laufen, Treppen	1	Decision Tree, Andere
[23]	-	Gehen, nicht-Gehen	2	Decision Tree, Naive Bayes, Support Vector Machine
[8]	Gyroskop	Liegen, Sitzen, Gehen, Laufen, in Bus	1	Decision Tree, Support Vector Machine, Andere
[24]	-	Stehen, Gehen, Laufen, Treppen	1	Decision Tree, Naive Bayes, Support Vector Machine, Andere
[25]	-	Gehen, Laufen/Joggen, Fahrradfahren, Springen	1	Andere
[26]	-	Stehen, Gehen, Laufen, Treppen	-	Decision Tree, Support Vector Machine, Andere
[27]	-	Gehen, Laufen, in Bus, in MRT (Metro in Singapur)	-	Decision Tree, Naive Bayes, Andere

[1] Es wurden dazu noch die Wechsel zwischen Bewegungen untersucht (Transitionen), wie z.B. Aufstehen, also dem Wechsel von Liegen zu Stehen oder Sitzen zu Stehen

Tabelle 1 zeigt eine Übersicht vorhandener Untersuchungen, bei denen die Bewegungserkennung nur mit Hilfe des Smartphones durchgeführt worden ist. Neben den angegebenen zusätzlichen Sensoren wurde immer der Beschleunigungssensor verwendet. Die Ansätze und Zwecke der Studien unterscheiden sich dabei in verschiedenen Punkten.

So sollte in Chicago die Bewegungserkennung für eine Standortverfolgung des Benutzers eingesetzt werden, damit dem Benutzer genaue Informationen über die öffentlichen Verkehrsmittel bereitgestellt werden können [3]. Um herauszufinden, ob und wo sich der Benutzer in einem öffentlichen Verkehrsmittel befindet, werden zwei Schritte durchgeführt: Als erstes werden Beschleunigungssensor und GPS für die Unterscheidung zwischen stationär und bewegend bezüglich des Benutzers verwendet. Im Falle einer Bewegung wird mit WiFi und wiederum GPS versucht, die eigentliche Standortverfolgung durchzuführen. Der erste Schritt ist der für diese Arbeit relevante, da Gehen erkannt werden und von nicht-Gehen unterschieden werden muss. Für die Auswertung wurde ein Decision Tree verwendet, der Gehen zu 97,5 % und nicht-Gehen zu 99,9 % erkennen kann.

In [23] wurde ebenfalls versucht, das Gehen eines Benutzers zu erkennen. Allerdings wurde ein anderer Ansatz verfolgt, bei dem zunächst die Position des Smartphones herausgefunden werden soll. Je nach Position werden die Daten dann anderes weiterverarbeitet. Da die Erkennung der Smartphone Position mit über 98 % sehr hoch ist, erreicht auch die darauffolgende Klassifizierung der Bewegung eine hohe Genauigkeit (über 98,4 % bei beiden verwendeten Positionen). Beide Untersuchungen betrachteten mit zwei bzw. drei nur relativ wenige Bewegungen.

Die anderen in Tabelle 1 aufgelisteten Untersuchungen betrachteten mindestens vier Bewegungen. Dabei wurde jedes Mal Gehen mit einbezogen. Weitere häufig untersuchte Bewegungen sind Sitzen, Stehen, Laufen, Treppensteigen und ob die Person in einem Fahrzeug ist. Je mehr ähnliche Bewegungen betrachtet werden, desto schwieriger ist die Erkennung, da z.B. beim Treppensteigen wenige Unterschiede zum normalen Gehen zu erwarten sind.

Ein weiterer wichtiger Punkt der Bewegungserkennung ist die Trainingsphase der Klassifizierungsalgorithmen, da die spätere Auswertung der Sensordaten auf deren Ergebnis - dem sogenannten Modell - basiert. [11] untersucht, inwiefern ein Modell von den Personen abhängt, die Trainingsdaten bereitgestellt haben. In zwei Sitzungen wurden von acht Personen Daten gesammelt. Der erste Datensatz wurde als Trainings- und der zweite als Testdatensatz verwendet. Um die Abhängigkeit des Benutzers in Bezug auf das Modell zu testen wurde einerseits für jeden Benutzer ein eigenes Modell auf Basis der jeweiligen Trainingsdaten (Within-Person Model) und andererseits ein Modell auf Basis der Trainingsdaten der anderen sieben Personen erstellt (Cross-Person Model). Das Cross-Person Model war mit 97,4 % nur etwas ungenauer als das Within-Person Model mit 99,48 %. Das zeigt, dass die Verwendung von Klassifizierungsalgorithmen gegen das Problem verschiedener Benutzer robust und schon mit einer geringen Anzahl an Testpersonen möglich ist. Allerdings wurde bei diesem Versuch nur eine Position des Smartphones betrachtet.

Verschiedene Positionen des Smartphones führen zu unterschiedlichen Werten der Sensoren bei gleichen Bewegungen [12]. Abbildung 1 zeigt diese Unterschiede bei der Bewegung Gehen. Es werden drei unterschiedliche Positionen gezeigt, bei denen die Messwerte des Beschleunigungssensors am deutlichsten bei der Position *Jeanstasche* ausgeprägt sind. Dies liegt daran, dass beim Gehen die Beine am meisten beschleunigt werden. Die in Tabelle 1 aufgezählten

Untersuchungen, bei denen die Position des Smartphones beliebig ist, verwendeten keine Restriktionen beim Tragen des Smartphones. Dieses Problem wurde also schon häufiger untersucht.

Kaum untersucht wurde der Nutzen weiterer Sensoren, die in Smartphones vorhanden sind. Alle aufgelisteten Untersuchungen benutzten den Beschleunigungssensor, der Großteil sogar ausschließlich. Neben dem Beschleunigungssensor wurde GPS am häufigsten verwendet. Mit GPS können neben der Position des Benutzers verschiedene weitere Informationen gesammelt werden, allerdings verbraucht GPS auch sehr viel Akkuleistung des Smartphones (vgl. Kapitel 2.1.6).

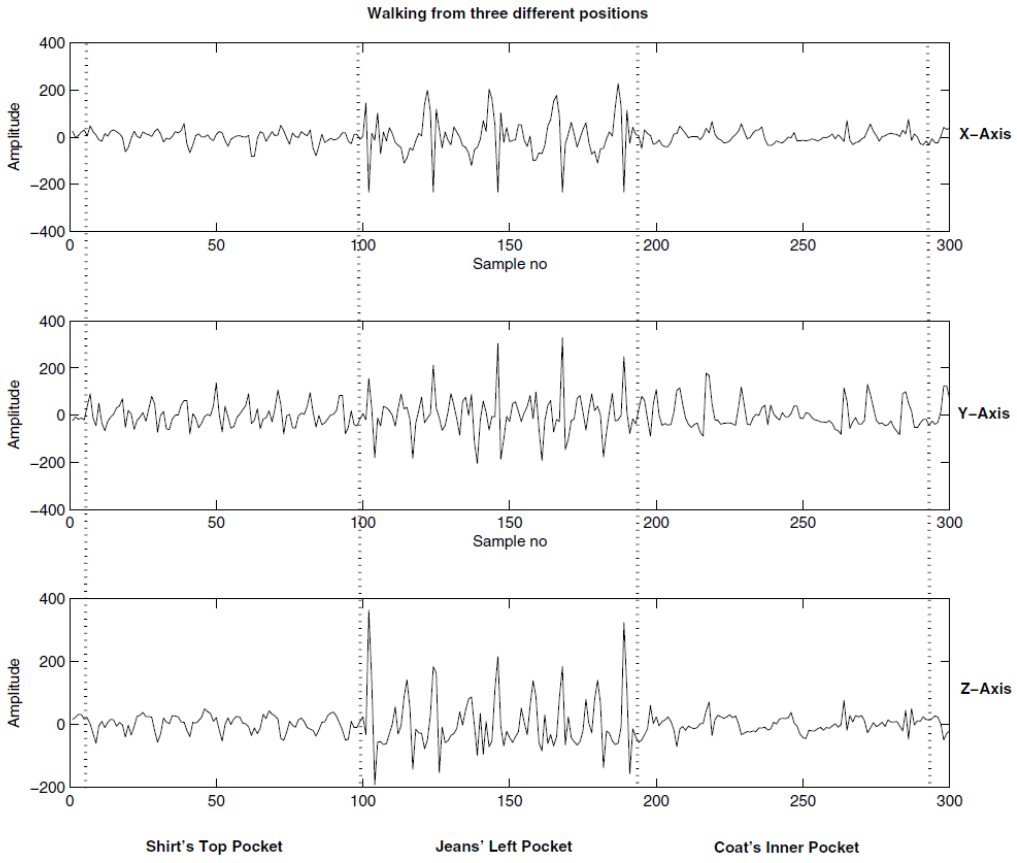

Abbildung 1 Sensorwerte eines dreiachsigen Beschleunigungssensor für Gehen bei drei verschiedenen Positionen, aus [12]

Ebenfalls wurde WiFi oder das Mikrofon in Untersuchungen verwendet. Allerdings sind diese Sensoren nicht direkt für die Bewegungserkennung eingesetzt worden, sondern hatten andere Funktionen. WiFi wurde in [3] neben GPS für die Standortverfolgung verwendet und in [14] sollte der Ort des Benutzers erkannt werden, indem die Wireless Access Points von häufig besuchten Orten gespeichert werden. Das Mikrofon wurde verwendet, um herauszufinden, ob der Benutzer in einer lauten oder ruhigen Umgebung ist. Mit dieser Information und den Daten des Beschleunigungssensors wurde dann automatisch entschieden, ob GPS aktiviert werden soll oder nicht. Das Mikrofon ist dementsprechend ebenfalls kein direkter Teil der Bewegungserkennung, sondern nur ein Auslöser für andere Sensoren. Auch der in [19] verwendete Nähesensor und die Kameraintensität wurden nicht für die Bewegungserkennung an sich, sondern für die Bestimmung der Position des Smartphones eingesetzt. Lediglich von dem (eindimensionalen) Gyroskop wurden in [8] die Sensordaten für die Bewegungserkennung verwendet. Die Daten dieses Sensors wurden wie die Daten des Beschleunigungssensors verwendet und z.B. Maximum, Minimum, Durchschnitt und

Varianz berechnet. Inwiefern die Sensordaten des Gyroskops die Bewegungserkennung beeinflusst haben ist jedoch nicht erwähnt worden.

Zusammengefasst wurde die Bewegungserkennung also schon zu großen Teilen untersucht und kann bereits mit ausschließlicher Hilfe des Smartphones durchgeführt werden. Das Smartphone kann dabei an beliebigen Positionen am Körper getragen werden und auch die Ausrichtung ist beliebig. Es können verschiedene Bewegungen erkannt werden, wobei Sitzen, Stehen, Gehen und Laufen am häufigsten untersucht wurde. Allerdings wurde fast ausschließlich der Beschleunigungssensor für die Bewegungserkennung verwendet, obwohl in Smartphones weitere Sensoren integriert sind, die ebenfalls nützlich sein könnten. Diese Arbeit untersucht somit den Nutzen weitere Sensoren, die mit dem Android 2.3.4 verfügbar sind.

1.4 Aufbau der Arbeit

Kapitel 2 befasst sich mit den Grundlagen, die für die hier umgesetzte Bewegungserkennung relevant sind. Dabei wird auf verfügbare Sensoren des Smartphones eingegangen (Kapitel 2.1) und die verwendeten Klassifizierungsalgorithmen erläutert (Kapitel 2.2).

Kapitel 3 beschreibt die konzeptionelle Umsetzung und begründet spezielle Entscheidungen bezüglich der Bewegungserkennung. Kapitel 3.1 beschreibt den allgemeinen Ablauf der Bewegungserkennung und Kapitel 3.2 bis 3.7 behandeln die genaueren Entscheidungen der Umsetzung, wie z.B. die Wahl der Sensoren und deren Abtastrate.

Nach der theoretischen und konzeptionellen Aufarbeitung wird die praktische Umsetzung beschrieben. Dafür werden verschiedene Softwarewerkzeuge verwendet, die in Kapitel 4 behandelt werden. Kapitel 5 befasst sich mit der praktischen Umsetzung, die in Datensammlung (Kapitel 5.1), Datenverarbeitung (Kapitel 5.2), Data Mining (Kapitel 5.3) und der eigentlichen Anwendung der Bewegungserkennung in einer Android Applikation unterteilt wird (Kapitel 5.4).

Um die Qualität der Bewegungserkennung zu testen, befasst sich Kapitel 6 mit den Ergebnissen und der Validierung der Modelle. Dafür werden Testdaten verwendet, die klassifiziert werden müssen.

Abschließend fasst Kapitel 7 diese Arbeit in Form eines Fazits zusammen und in Kapitel 8 werden Anmerkungen für zukünftige Arbeiten gemacht.

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der für die Bewegungserkennung relevanten Aspekte. Kapitel 2.1 beschreibt dabei die Sensoren, die das Android Framework zur Verfügung stellt.

In Kapitel 2.2 werden drei verschiedene Klassifizierungsalgorithmen vorgestellt, die oftmals in der Literatur verwendet werden und die auch diese Arbeit benutzen wird.

2.1 Sensoren in Android Smartphones

Die Sensoren sind die Grundlage der Bewegungserkennung, da sie Informationen liefern, die eine Erkennung erst möglich machen. Der Beschleunigungssensor misst für jede Achse des Smartphones die Beschleunigung, wobei verschiedene Abtastraten möglich sind. Wenn das Smartphone in der vorderen Hosentasche des Benutzers liegt, wird es bei jedem Schritt durch die Bewegung des Beines beschleunigt. Diese Beschleunigung wird gemessen und anhand dieser Werte ist es möglich, die Bewegung zu erkennen. Abbildung 2 zeigt einen zehn Sekunden Ausschnitt der Bewegung Gehen, der mit dem Beschleunigungssensor aufgezeichnet worden ist. Die genauere Verarbeitung mit diesen Sensorwerten wird in Kapitel 3.7 beschrieben.

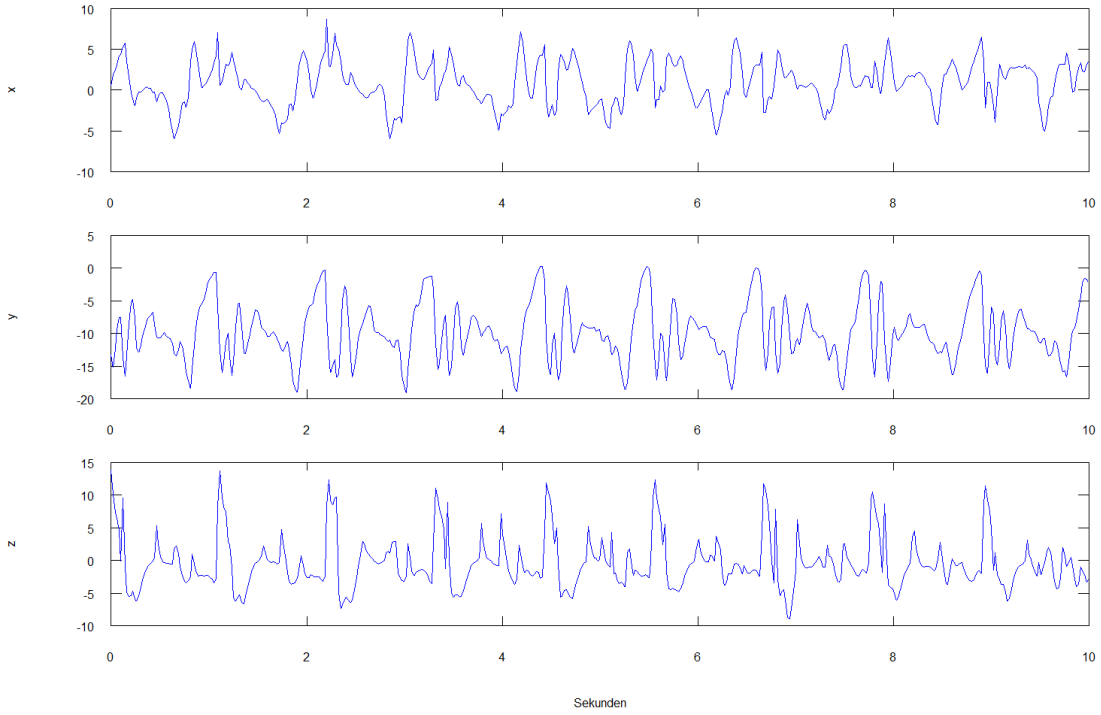

Abbildung 2 Zehn Sekunden Ausschnitt der Messwerte des Beschleunigungssensor der Bewegung Gehen

Verschiedene Sensoren sind über das Android Betriebssystem verfügbar [28]. Die Sensoren können mit hoher Präzision Rohdaten messen und liefern so Informationen über Bewegung und Position des Smartphones. Android stellt mit Hilfe der API Schnittstellen zur Verfügung, um die Sensoren des Smartphones nutzen zu können. Hinter manchen Sensoren steht ein echter Sensor im technischen Sinne, d.h. ein Stück Hardware, das die Messung durchführt. Andere Sensoren können nur auf Software basieren. Bei diesen steht kein echter Sensor dahinter, sondern die Werte werden von anderen echten Sensoren abgeleitet. Insgesamt werden 14 verschiedene Sensoren von der Android Plattform unterstützt. Tabelle 2 zeigt eine Übersicht über diese Sensoren, ob es sich um echte oder abgeleitete Sensoren handelt (Hardware oder Software) und die Kategorie, in die man den Sensor einordnen kann. Ein weiterer Punkt der Tabelle ist das verwendete Koordinatensystem der Sensoren. Während die Umgebungssensoren einfache Messwerte ausgeben, wie z.B. die Temperatur in Grad Celsius, benötigen die anderen Sensoren ein Koordinatensystem, in dessen Relation die Messwerte zu verstehen sind. Abbildung 3 zeigt das Koordinatensystem relativ zum Gerät und Abbildung 4 das Koordinatensystem relativ zur Erde. Die letzte Spalte von Tabelle 2 beschreibt die Verfügbarkeit der Sensoren in verschiedenen Android Versionen. Bei älteren Android Versionen sind manche Sensoren nicht verfügbar.

Tabelle 2 Sensortypen der Android Plattform und deren Verfügbarkeit, aus [29]

Sensor	Sensortyp	Kategorie	Verwendetes Koordinatensystem	Kurzbeschreibung	Verfügbar seit Android Version / API Level
Beschleunigungssensor	Hardware	Bewegungssensor	Relativ zum Gerät	Misst die Beschleunigung des Geräts in m/s^2 (inkl. Schwerkraft)	1.5 / 3
Druck	Hardware	Umgebungssensor	-	Misst den Luftdruck in hPa oder mbar	2.3 / 9
Gyroskop	Hardware	Bewegungssensor	Relativ zum Gerät	Misst die Rotationsrate in rad/s	2.3 / 9
GPS	Hardware	Positionssensor	-	Misst Position des Smartphones in Form von Longitude und Latitude	-
Licht	Hardware	Umgebungssensor	-	Misst das Lichtlevel in lx	1.5 / 3
Lineare Beschleunigung	Software oder Hardware	Bewegungssensor	Relativ zum Gerät	Misst die Beschleunigung des Geräts in m/s^2 (exkl. Schwerkraft)	2.3 / 9
Magnetisches Feld (Kompass)	Hardware	Positionssensor	Relativ zum Gerät	Misst das geomagnetische Feld in µT	1.5 / 3
Nähe	Hardware	Positionssensor	-	Misst den Abstand eines Objektes relativ zum Bildschirm in cm	1.5 / 3
Orientierung	Software	Positionssensor	Relativ zur Erde	Misst den Rotationsgrad des Gerätes	1.5 / 3 (veraltet)
Relative Luftfeuchtigkeit	Hardware	Umgebungssensor	-	Misst die relative Luftfeuchtigkeit in Prozent (%)	4.0 / 14
Rotationsvektor	Software oder Hardware	Bewegungssensor	Relativ zur Erde	Misst die Rotation des Geräts (einheitslos)	2.3 / 9
Schwerkraft	Software oder Hardware	Bewegungssensor	Relativ zum Gerät	Misst die Schwerkraft, die auf das Gerät wirkt in m/s^2	2.3 / 9
Temperatur	Hardware	Umgebungssensor	-	Misst die Temperatur des Gerätes in Grad Celsius (°C)	1.5 / 3 (seit Android 4.0 abgelöst durch Sensor *Umgebungstemperatur*)
Umgebungstemperatur	Hardware	Umgebungssensor	-	Misst die Raumtemperatur in Grad Celsius (°C)	4.0 / 14

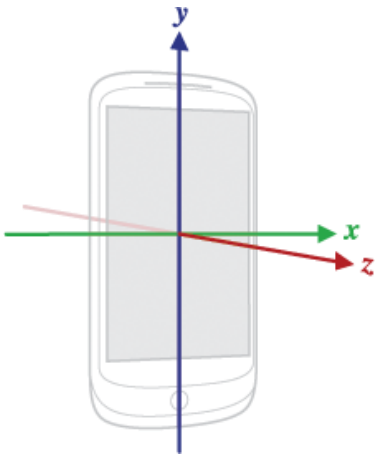

Abbildung 3 Koordinatensystem relativ zum Gerät, aus [31]

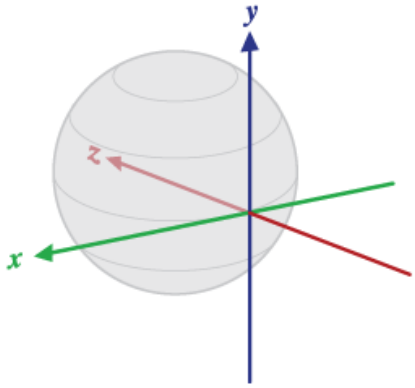

Abbildung 4 Koordinatensystem relativ zur Erde, aus [31]

Die Applikation der Bewegungserkennung wird mit Android 2.3.4 umgesetzt, die zu der Android Version 2.3 (API Level 9) gehört. Demnach sind alle Sensoren außer der relativen Luftfeuchtigkeit und der Umgebungstemperatur verfügbar. Von diesen beiden Sensoren sind ohnehin keine relevanten Informationen zu erwarten. Zu jeder Android-Version gehört eine Komptabilitätsdefinition, die Anforderungen an Geräte stellt, die eine bestimmte Android-Version benutzen [30]. Vorhandene Einträge zu den hier dargestellten Sensoren werden bei dem jeweiligen Sensor erwähnt.

Die für die Bewegungserkennung relevanten Sensoren werden in den Kapiteln 2.1.1 bis 2.1.6 genauer untersucht. Relevant sind die Sensoren, die den Kategorien Positionssensor und Bewegungssensor zugeordnet werden. Umgebungssensoren bieten keine nützlichen Informationen, da Temperatur oder Luftfeuchtigkeit keinen Aufschluss auf die durchgeführte Bewegung ermöglichen. Auch der Positionssensor *Nähe* wird nicht verwendet, da dieser die Nähe zu einem Objekt misst, was ebenfalls keinen Aufschluss auf die durchgeführte Bewegung ermöglicht. Dieser Sensor wird z.B. verwendet, um zu überprüfen, ob der Benutzer das Smartphone am Ohr hält, also am Telefonieren ist. Für die Bewegungserkennung wurde dieser Sensor wie bereits erwähnt in [19] verwendet, um die Position des Smartphones herauszufinden.

Die relevanten Sensoren werden soweit möglich auf verschiedene Kriterien hin untersucht und hinsichtlich der Verwendung für die Bewegungserkennung bewertet. Diese Kriterien sind:

- Funktion
- Verbrauch
- Alternativen
- Einschränkungen
- Verwendungsgrund
- Vor- und Nachteile
- Kompatibilität mit Android Geräten

Die Sensoren Beschleunigung, Schwerkraft und lineare Beschleunigung werden zusammen in Kapitel 2.1.1 betrachtet, da sie in folgendem Zusammenhang stehen:

Beschleunigung = Lineare Beschleunigung + Schwerkraft

Außerdem ist in vielen Endgeräten nur der Beschleunigungssensor ein echter Sensor, während Schwerkraft und lineare Beschleunigung daraus abgeleitet werden (vgl. Tabelle 2).

Wenn nicht anders vermerkt, stammen die folgenden Informationen von der Android Developer Webseite [31] und die Informationen über die Kompatibilität von [30].

2.1.1 Beschleunigungssensor

Funktion

Der Beschleunigungssensor berechnet für jede der drei Achsen des Smartphones die momentane Beschleunigung in m/s². Das Koordinatensystem ist relativ zum Gerät (vgl. Abbildung 3).
Die Schwerkraft beeinflusst die Messungen des Sensors. Daher wird der Sensor bei einem Smartphone, das auf dem Tisch liegt, je nach Ausrichtung einen Wert von 9,81 m/s² oder -9,81 m/s² auf der z-Achse anzeigen [31].

Verbrauch

Der Beschleunigungssensor verbraucht nur wenig Energie bei den Messungen [18] und benötigt vor allem im Vergleich mit anderen Sensoren viel weniger Energie. GPS benötigt ungefähr fünf bis sieben Mal so viel Energie bei dauerhafter Verwendung [14] [32], das Mikrofon ungefähr vier bis fünf Mal, ein WiFi Scan ungefähr 23 bis 24 Mal und ein Bluetooth Scan ungefähr drei bis vier Mal so viel Energie [32].

Alternativen

Eine Alternative ist die lineare Beschleunigung, die gleiche Werte liefert, nur dass die Schwerkraft herausgerechnet wird. Andere Sensoren, wie z.B. das Gyroskop, messen zwar andere Werte, aber mit diesen können Bewegungen ebenfalls erkannt werden. Inwieweit die Sensoren besser geeignet sind, wird in Kapitel 6 beschrieben.

Einschränkungen

Je nach Smartphone gibt es eine maximale Abtastrate. Die Kompatibilitätsdefinition schreibt eine minimale Abtastrate von 50 Hz vor, aber falls eine höhere Abtastrate nötig ist, kann diese nicht garantiert werden. Außerdem werden Messungen nicht in den exakt gleichen Zeitabständen durchgeführt, was zu Unterschieden in der Anzahl der Messungen pro Sekunde führen kann. In eigenen Versuchen wurden bei einer Einstellung von 50 Hz zwischen 46 und 50 Werten pro Sekunde gemessen.

Verwendungsgrund

Vergangene Untersuchungen haben gezeigt, dass mit Hilfe des Beschleunigungssensors Bewegungen erkannt werden können, da durch die Bewegungen des Benutzers das Smartphone in der Hosentasche charakteristisch beschleunigt wird.

Vorteile

Der Beschleunigungssensor kann jederzeit verwendet werden und ist beim Messen zuverlässig – z.B. kann GPS hingegen ausfallen. Wie der Punkt *Verbrauch* gezeigt hat, ist dieser Sensor auch akkuschonend und kann daher auch über einen längeren Zeitraum verwendet werden.

Nachteile

Die unbekannte Position des Smartphones kann zu einer signifikanten Veränderung der absoluten Werte (vgl. Abbildung 1, S. 11) und somit auch der in den Frequenzbereich umgerechneten Werte führen [3].

Das Sammeln der Sensordaten allgemein ist ein Risiko für die Privatsphäre und können ein Sicherheitsproblem darstellen Beispielsweise könnten Eingaben der Tastatur mit Hilde der Sensordaten erkannt werden [33].

Kompatibilität

Die Kompatibilitätsdefinition sieht vor, dass ein Gerät ein drei-Achsen Beschleunigungssensor implementieren sollte. Falls vorhanden, muss es:

- mindestens 50 Hz leisten
- das Android Sensor Koordinatensystem verwenden (Abbildung 3)
- fähig sein, vom freien Fall bis zu zweifacher Schwerkraft (2 g) oder mehr messen zu können (für jede Achse)
- eine Genauigkeit von mindestens 8 Bits haben
- eine Standardabweichung haben, die geringer als 0,05 m/s² ist

Diese Anforderungen sind für die Bewegungserkennung ausreichend, auch wenn ein Beschleunigungssensor keine Pflicht ist.

2.1.2 Orientierung

Funktion

Dieser Sensor berechnet die Orientierung des Smartphones relativ zur Erde (vgl. Abbildung 4). Die Messung des Sensors wird in drei Werte geteilt: *Azimut*, *Pitch* und *Roll*. Diese drei Bezeichnungen beschreiben die Rotationen um die drei Achsen, wobei Azimut der Rotation um die Z-Achse, Pitch um die Y-Achse und Roll um die X-Achse entspricht.

Im Gegensatz zum Beschleunigungssensor heißt das auch, dass sich die absoluten Werte je nach Bewegungsrichtung des Benutzers ändern.

Alternativen

Ähnlich funktioniert der Sensor des Magnetfeldes, bei dem sich lediglich das gezeigte Bezugssystem ändert, siehe Kapitel 2.1.3.

Einschränkungen

Wie schon bei dem Beschleunigungssensor gibt es eine Obergrenze der Abtastrate. Außerdem werden auch hier die Werte nicht in genau gleichen Abständen gemessen.

Dazu kommt, dass die Orientierung kein echter Sensor ist und mit Hilfe der Schwerkraft und des geomagnetischen Feldes berechnet wird. Daher gelten dieselben Einschränkungen wie auch für den Sensor des magnetischen Feldes, der in Kapitel 2.1.3 behandelt wird.

Verwendungsgrund

Wird zu Testzwecken verwendet um zu überprüfen, ob mit diesem Sensor Bewegungen erkannt werden können.

2.1.3 Magnetisches Feld

Funktion

Misst das magnetische Feld in micro-Tesla (uT) und somit die Ausrichtung des Geräts in Bezug auf das magnetische Feld der Erde. Es hat eine Komponente, die zum magnetischen Nordpol zeigt, was als Kompass verwendet werden kann [34].

Alternativen

Ähnlich funktioniert der Sensor der Orientierung, bei dem sich lediglich das gezeigte Bezugssystem ändert, siehe Kapitel 2.1.2.

Einschränkungen

Der Sensor ist empfindlich gegen verschiedene Sorten metallischer Objekte. Außerdem kann bei einer horizontalen Ausrichtung des Smartphones die Messung verfälscht werden [34].

Verwendungsgrund

Wird zu Testzwecken verwendet um zu überprüfen, ob mit diesem Sensor Bewegungen erkannt werden können.

Nachteile

Die absoluten Werte ändern sich je nach Richtung, in die das Smartphone ausgerichtet ist. Daher ist z.B. der Mittelwert über einen kleinen Zeitraum nicht aussagekräftig.

Kompatibilität

Die Kompatibilitätsdefinition sieht vor, dass ein Gerät ein drei-Achsen Magnetometer (z.B. Kompass) implementieren sollte. Falls vorhanden, muss es:

- mindestens 10 Hz leisten
- das Android Sensor Koordinatensystem verwenden (Abbildung 3)
- fähig sein, eine Feldstärke zu messen, die für das geomagnetische Feld ausreichend ist
- eine Genauigkeit von mindestens 8 Bits haben
- eine Standardabweichung haben, die geringer als $0{,}5\ \mu T$ ist

2.1.4 Rotationsvektor

Funktion

Misst den Rotationsvektor um die drei Achsen und optional die Skalarkomponente des Rotationsvektors. Dieser Sensor ist ideal für die Erkennung von Gesten und Überwachung der Ausrichtung.

Alternativen

Beschleunigungssensor, magnetisches Feld und Orientierung können als Alternative verwendet werden.

Verwendungsgrund

Wird zu Testzwecken verwendet um zu überprüfen, ob mit diesem Sensor Bewegungen erkannt werden können.

2.1.5 Gyroskop

Funktion

Das Gyroskop gehört zu den Sensoren, die Werte relativ zum Gerät aufnehmen. In diesem Fall wird die Winkelgeschwindigkeit um die drei Achsen gemessen.

Verwendungsgrund

Das Gyroskop ist ein weiterer Sensor mit dem gleichen Koordinatensystem wie der Beschleunigungssensor, aber misst andere Werte, über die eine Bewegung womöglich ebenfalls erkannt werden kann.

Kompatibilität

Die Kompatibilitätsdefinition sieht vor, dass ein Gerät ein Gyroskop implementieren sollte. Geräte sollten kein Gyroskop enthalten, wenn ein drei-achsiger Beschleunigungssensor enthalten ist. Falls ein Gyroskop vorhanden ist, muss es:

- mindestens 100 Hz leisten
- fähig sein, Wechsel in der Ausrichtung bis zu 5,5*Pi Bogenmaß/Sekunde zu berechnen
- eine Genauigkeit von mindestens 8 Bits haben

Nachteile

Falls die Kompatibilitätsdefinition von Herstellern strikt befolgt wird, kann das Gyroskop nicht mit dem Beschleunigungssensor verwendet werden, da bei Existenz eines Beschleunigungssensors kein Gyroskop vorhanden sein sollte.

2.1.6 GPS

Funktion

Durch GPS kann die genaue Position des Users erkannt werden. Genauer gesagt lässt sich die geographische Länge (Longitude), geographische Breite (Latitude) und die Höhe (Altitude) ermitteln. Durch den Abgleich zweier Positionen und der verstrichenen Zeit lässt sich so die Durchschnittsgeschwindigkeit berechnen, mit der sich das Smartphone fortbewegt hat. Für die Geschwindigkeit stellt das Android Framework eine einfache Methode bereit.

Verbrauch

Bis zu fünf Mal höher als der Beschleunigungssensor (vgl. Kapitel 2.1.1)

Alternativen

Theoretisch lässt sich die Geschwindigkeit auch über den Beschleunigungssensor berechnen.

Einschränkungen

Kann in Gebäuden und im Untergrund (z.B. in U-Bahnen) nicht mit Sicherheit verwendet werden, da der Empfang gestört sein oder komplett ausfallen kann. Auch bei besonderen Wetterlagen kann es zu Störungen kommen, wie z.B. bei einem Schneesturm oder Gewitter,.

Verwendungsgrund

Für die Bewegungserkennung ist die eigentliche Position nicht unbedingt relevant, da es unwichtig ist, wo der User z.B. am Laufen ist. Der relevante Aspekt ist die Geschwindigkeit, mit der sich der User fortbewegt.

Vorteile

Genaue Bestimmung der Geschwindigkeit des Benutzers.

Nachteile

Siehe Verbrauch und Einschränkungen. Dazu kommt, dass GPS nur mit Genehmigung des Users benutzt werden kann. Aufgrund des hohen Verbrauchs sollte GPS auch nicht dauerhaft laufen, so dass nicht jede Sekunde die Geschwindigkeit vorliegen könnte. GPS sollte nur verwendet werden, wenn es unbedingt notwendig ist.

Kompatibilität

Die Kompatibilitätsdefinition sieht vor, dass ein Gerät einen GPS Empfänger implementieren sollte. Falls vorhanden, sollte es eine Art „assistierender GPS“ Technik beinhalten.

2.2 Klassifizierungsalgorithmen

Ein Klassifizierungsalgorithmus wird dem Data Mining zugeordnet und kann Klassifizierungsprobleme lösen. [35, p. 1] definiert Data Mining als eine Wissenschaft und Technologie von Datenuntersuchungen, um vorher unbekannte Muster in Daten zu finden. Die Paradigmen des Data Minings lassen sich in zwei Kategorien unterteilen (Abbildung 5): Verifikation und Entdeckung. Bei der Verifikation geht es um das Testen von erstellten Modellen des Data Minings während die Entdeckung versucht, Muster in Daten zu entdecken. Die drei Klassifizierungsalgorithmen *Decision Tree*, *Naive Bayes* und *Support Vector Machine*, die in dieser Arbeit verwendet werden, sind solche Klassifizierungsalgorithmen und können innerhalb der Entdeckung der Kategorie Vorhersage zugeordnet werden, da es um die Erstellung von Modellen geht, die weitere Daten erkennen und somit sozusagen hervorsagen sollen.

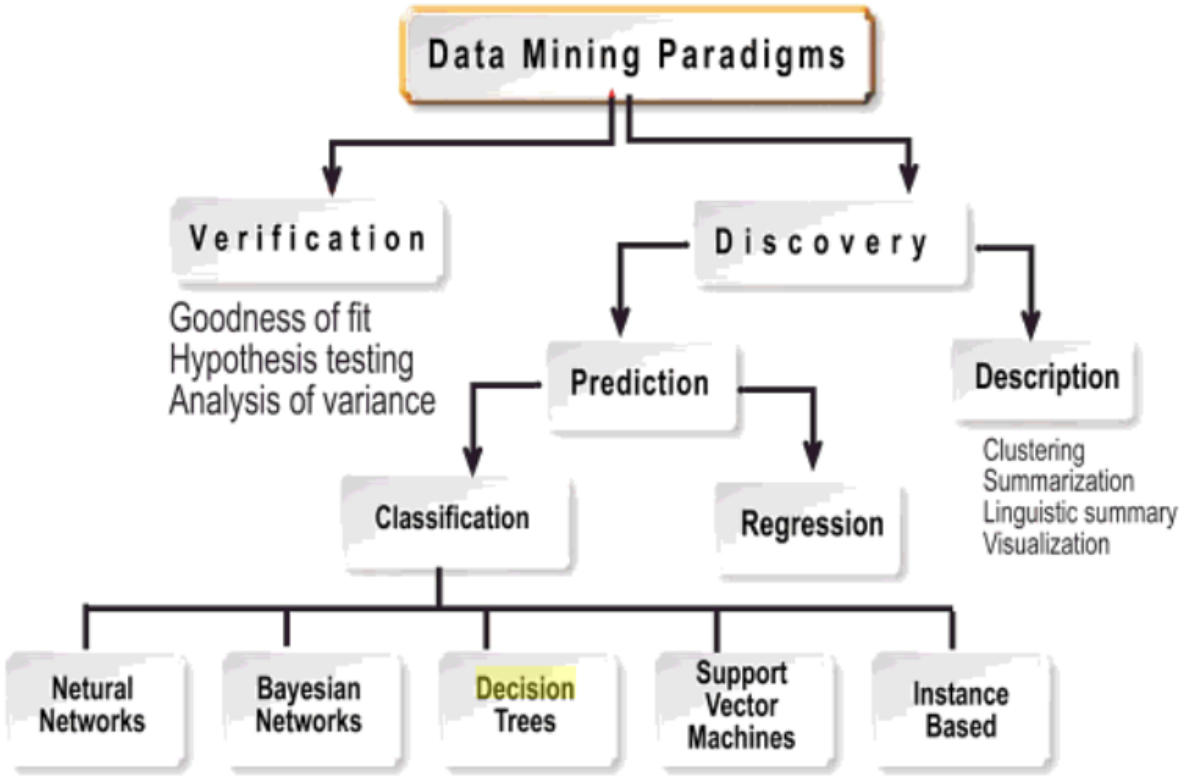

Abbildung 5 Data Mining Paradigmen, aus [35] Kapitel 1.1

Die Klassifizierung bezeichnet die Zuordnung von beliebig vielen unterschiedlichen Attributen zu einer bestimmten Kategorie. Jedes Attribut kann die Zuordnung beeinflussen. Ein einfaches Beispiel in Anlehnung an [36] ist die Entscheidung, ob man spazieren gehen soll oder nicht. Die Kategorie ist dabei *Ja* oder *Nein*. Mögliche Attribute, die für diese Entscheidung relevant sind, können das Wetter oder die Temperatur sein. Damit eine Klassifizierung funktioniert, wird eine Trainingsphase benötigt, in der dem Klassifizierungsalgorithmus beigebracht wird, wie er sich zu verhalten hat. So kann z.B. schon vorher bekannt sein, dass man spazieren geht, wenn die Sonne scheint (Wetter) und es 25 Grad sind (Temperatur). Je nach Klassifizierungsalgorithmus wird die Klassifizierung neuer Werte anders ermittelt. Diese werden in den folgenden Kapiteln anhand dieses Beispiels erklärt.

2.2.1 Naive Bayes

Naive Bayes ist ein auf Wahrscheinlichkeiten ruhender Klassifizierungsalgorithmus. Es basiert auf der Annahme, dass einzelne Attribute voneinander unabhängig und nur vom Klassenattribut abhängig sind [37]. Obwohl Naive Bayes einfach aufgebaut ist, hat es eine hohe Performanz. Das ist auch in Bereichen der Fall, bei denen es klare Abhängigkeiten zwischen den Attributen gibt. Dies ist für die Bewegungserkennung möglicherweise von Relevanz, da z.B. die Sensoren der Beschleunigung zusammenhängen. Naive Bayes bestimmt die Zugehörigkeit eines zu klassifizierendes Wertes mit Hilfe der bereits vorhandenen Daten und berechnet die Wahrscheinlichkeit über die Häufigkeit der Zugehörigkeit. Sollte bezüglich des Beispiels ein neuer Wert mit den Attributen sonnig und 20 Grad klassifiziert werden, so sucht Naive Bayes die Kategorien der jeweiligen einzelnen Attribute. Sollte unabhängig von anderen Attributen das Wetter Sonne fünf Mal als *Ja* und drei Mal als *Nein* kategorisiert worden sein, so wäre die Wahrscheinlichkeit 5/8, also 62,5 %, dass dieses Attribut und damit der gesamte Wert zu der Kategorie *Ja* gehört. Die endgültige Wahrscheinlichkeit wird aus den Wahrscheinlichkeiten aller Attribute berechnet.

2.2.2 Decision Tree

Ein Decision Tree beschreibt einen Baum, in dessen Blättern der zu klassifizierende Wert steht [38]. Der Weg zu diesen Blättern führt über verschiedene Knoten, auf denen ein Vergleich mit einem Attribut durchgeführt wird. Ein solcher Vergleich könnte mit dem Attribut *Temperatur* stattfinden. Falls die Temperatur unter fünf Grad liegt, kann der Wert direkt als *Nein* klassifiziert werden. Bei einer höheren Temperatur können weitere Vergleiche folgen, die sich dann auf das Attribut *Wetter* beziehen.

Für die Bewegungserkennung ist der Decision Tree verwendbar, da es sehr viele numerische Attribute geben kann und der Decision Tree mit Hilfe von Schwellenwerten eine Klassifizierung sehr robust machen kann.

2.2.3 Support Vector Machine

Support Vector Machines sind eine Mischung aus linearer Modellierung und Instanz basiertem Lernen [38]. Wie bei dem Decision Tree wird versucht, die Kategorien zu separieren. Jeder Eintrag einer Kategorie wird als Vektor betrachtet, der unendlich viele Attribute haben kann. Die Grenzen jeder Kategorie werden als *Support Vector* gezeichnet, da über diese eine Funktion erstellt wird, die Kategorien teilt. Diese Funktion ist im einfachsten Fall linear, muss aber keine bestimmte Form haben. In Bezug auf das Beispiel könnte eine mögliche Funktion eine Linie sein, welche die Temperatur bei fünf Grad teilt, so dass alle Werte darunter zu der Kategorie *Nein* gehören.

3 Konzeptionelle Umsetzung

Dieses Kapitel beschreibt die konzeptionelle Umsetzung der Bewegungserkennung. Dabei wird zunächst der Ablauf der Bewegungserkennung beschrieben (Kapitel 3.1) und genauere Informationen zu einzelnen Punkten in den Unterkapiteln behandelt. Dazu gehört die Wahl der Sensoren (Kapitel 3.2), Unterschiede bei verschiedenen Benutzern (Kapitel 3.3), mögliche Positionen des Smartphones (Kapitel 3.4), Abtastrate der Sensoren (Kapitel 3.5), Zeitfenster (Kapitel 3.6) und als letztes die genauere Verarbeitung der Sensordaten (*Preprocessing*, Kapitel 3.7).

3.1 Ablauf der Bewegungserkennung

Der Ablauf kann in drei grundlegende Schritte eingeteilt werden, die in vielen vorherigen Untersuchungen vorhanden waren (Abbildung 6). Dazu gehört als erster Schritt die Messung und Speicherung von Sensordaten. Da ein Klassifizierungsalgorithmus nicht direkt mit den rohen aufeinander folgenden Messwerten der Sensoren umgehen kann, müssen diese in Zeitfenster eingeteilt werden [22]. Dieser zweite Schritt wird oft als Preprocessing bezeichnet, also die Vorverarbeitung der Daten. Außerdem werden die Daten so umgewandelt und verarbeitet (Kapitel 3.7), dass sie als Input für den dritten Schritt verwendet werden können. Dieser dritte Schritt ist die Verwendung eines der in Kapitel 2.2 beschriebenen Klassifizierungsalgorithmen, der die Bewegung klassifizieren soll. Dieser Ablauf ist z.B. in [1] oder [22] verwendet worden. Andere Untersuchungen versuchten vor der Verwendung des Klassifizierungsalgorithmus die Position des Smartphones herauszufinden, damit für jede Position eine spezielle Weiterverarbeitung der Daten verwendet werden konnte [19] [20] [23].

[3] verwendete den Beschleunigungssensor zunächst auf 1 Hz um herauszufinden, ob sich der Benutzer am Bewegen ist oder nicht. Erst wenn erkannt wird, dass sich der Benutzer in Bewegung befindet, wird die Abtastrate auf 20 Hz erhöht und GPS aktiviert. Dies spart Akkuleistung des Smartphones, wenn der Benutzer sich eine längere Zeit nicht bewegt. Allerdings startet die Bewegungserkennung somit auch zeitverzögert und ein oder mehrere Zeitfenster gehen verloren. Für diese Arbeit werden die drei beschriebenen Schritte umgesetzt.

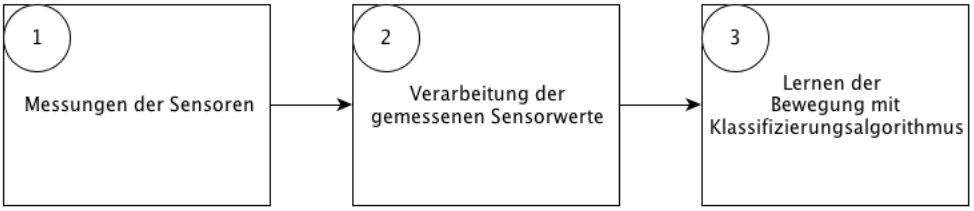

Abbildung 6 Grober Ablauf der Bewegungserkennung

3.2 Verwendete Sensoren

Kapitel 2.1 hat verschiedene Sensoren beschrieben, die für die Bewegungserkennung verwendet werden können. Da nur sehr wenige Untersuchungen andere Sensoren untersucht haben, werden alle Sensoren außer GPS betrachtet. Wie bereits erwähnt führt GPS zu einem erhöhten Akkuverbrauch und kann nicht überall eingesetzt werden (vgl. Kapitel 2.1.6).

Die folgenden Sensoren werden verwendet:

- Beschleunigungssensor
- Lineare Beschleunigung
- Schwerkraft
- Rotationsvektor
- Magnetisches Feld
- Gyroskop

Welche Werte aus den rohen Sensordaten ermittelt werden, beschreibt Kapitel 3.7.

3.3 Unterschiede bei verschiedenen Benutzern

Jeder Benutzer hat eine eigene Art, wie er sich bewegt. Daher ist es wichtig, mit welchen Daten der Klassifizierungsalgorithmus in der Trainingsphase erstellt wird. Zu unterscheiden sind Trainingsdaten, mit denen ein Modell erstellt wird und Testdaten, welche die Genauigkeit des Modells überprüfen. Wie bereits in Kapitel 1.3 erwähnt, kann die Bewegungserkennung auch dann erfolgen, wenn keine Daten des Benutzers, für den eine Bewegung erkannt werden soll, als Trainingsdaten verwendet worden sind. Dies wurde z.B. in [27] für Naive Bayes gezeigt. Daher wird diese Arbeit nur Testdaten von Benutzern benutzen, die nicht als Trainingsdaten im Modell enthalten waren.

3.4 Positionen des Smartphones

Unterschiedliche Positionen des Smartphones ändern sowohl die berechneten absoluten Werte als auch die Werte im Frequenzbereich [3]. Die unbekannte Position des Smartphones bei der späteren Erkennung der Bewegung könnte daher ein Problem darstellen, weil sich die Werte bei gleichen Bewegungen unterscheiden können. Beispielsweise kann es zu Ähnlichkeiten zwischen Laufen und Gehen kommen, wenn sich das Smartphone beim Gehen in der Hosentasche und beim Laufen in der Brusttasche befindet. Theoretisch ist dies denkbar, weil Laufen eine dynamischere Bewegung ist als Laufen, aber das Smartphone aufgrund der Position sich weniger bewegt. Um die Vergleichbarkeit der Ergebnisse zu erhalten und die Untersuchung auf die Verwendung verschiedener Sensorwerte zu konzentrieren, wurde sichergestellt, dass die Testpersonen die Smartphones in einer der beiden vorderen Hosentaschen tragen.

3.5 Abtastrate der Sensoren

Das Maximum der Abtastrate der Sensoren hängt von der Möglichkeit des Smartphones ab. Die Android-App Funf (vgl. Kapitel 4.1.2) zeichnet mit dem Beschleunigungssensor des Samsung Galaxy S2 bis zu 50 Werte pro Sekunde auf. Das Nexus 1 kann maximal 33,13 Hz und das Sony-Ericsson x10 mini maximal 96,44 Hz messen [34]. In vergangenen Untersuchungen wurden verschiedene Abtastraten verwendet. Diese reichen von 1 Hz (im Ruhemodus, [3]) bis zu 100 Hz [9] [13].

Das Android-Framework stellt keine Möglichkeit bereit, mit der man eine benutzerdefinierte Abtastrate festlegen kann, sondern bietet vier unterschiedliche Kategorien an. Diese unterscheiden sich je nach Smartphone. Da das Samsung Galaxy S2 verwendet wird, wurde eine Kategorie mit einer Abtastrate von 50 Hz gewählt. Da die Kompatibilitätsdefinition für Android 2.3.4 mindestens 50 Hz vorsieht, ist dieser Wert mit allen definitionskonformen Geräten kompatibel.

3.6 Zeitfenster

Da für die Bewegungserkennung ein gewisses Minimum an Daten vorliegen muss, werden die Daten in einem Zeitfenster gesammelt und erst nach wenigen Sekunden ausgewertet. Die Länge dieses Zeitfensters ist wichtig, da ein zu kurzes Zeitfenster womöglich nicht genügend Daten für eine korrekte Erkennung bereitstellt und ein zu langes Zeitfenster bei schnell wechselnden Bewegungen die Erkennung verfälscht.

Im Falle eines Zeitfensters von zwei Sekunden würde auch eine Verzögerung zwischen Bewegung und Erkennung von zwei Sekunden auftreten. Bei einer Aufzeichnung der durchgeführten Bewegungen über den Tag hinweg und späterer Auswertung wäre dies kein Problem, aber bei einer Live-Erkennung ist eine geringe Verzögerung wichtig.

In vorherigen Untersuchungen variiert die Länge des Zeitfensters zwischen einer Sekunde (z.B. [21] [19] [16]) und zehn Sekunden [22]. Das Zeitfenster von einer Sekunde erklärt [16] dadurch, dass eine Sekunde ausreichend für einen Fußschritt ist. Die optimale Länge des Zeitfensters hängt von der Bewegung ab [39], für alle hier untersuchten Bewegungen empfiehlt sich eine Länge von zwei Sekunden. Bei einer Abtastrate von 50 Hz und zwei Sekunden Zeitfenster ergeben sich also 100 Werte pro Zeitfenster.

Es wurde eine Überlappung der Zeitfenster gewählt, weil so mehrere Zeitfenster betrachtet werden. Dabei wurde eine Überschneidung von 50 % in vorherigen Untersuchungen vorgeschlagen [26]. Bei einem Zeitfenster von zwei Sekunden ergibt sich also eine Überschneidung von einer Sekunde.

Abbildung 7 zeigt das daraus resultierende Modell. Für jedes einzelne Zeitfenster können bestimmte Werte berechnet werden, die aus dem Zeit- oder Frequenzbereich stammen.

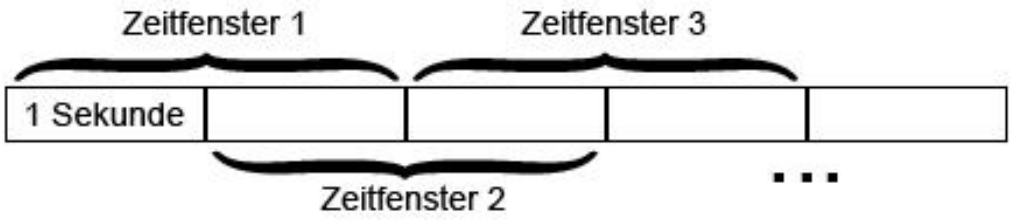

Abbildung 7 Modell der Zeitfenster

3.7 Preprocessing

Das Preprocessing beschreibt den Umgang mit den erfassten, rohen Sensordaten, die in den eben beschriebenen Zeitfenstern vorliegen. Die Daten müssen so verarbeitet werden, dass sie als Input für die Klassifizierungsalgorithmen verwendet werden können.

Das Preprocessing wurde in vergangenen Untersuchungen verschieden umgesetzt. Häufig wurde unter anderem der Mittelwert und die Standardabweichung für jede Achse des Beschleunigungssensors berechnet, wie z.B. in [2], [3], [15] oder [24]. In [14] wurde sogar ausschließlich die Standardabweichung des Beschleunigungssensors verwendet, da dieser Wert unabhängig von der Position ist. Erkannt werden sollte *Ruhen*, *Gehen*, *Laufen* und *in Fahrzeug*. *Ruhen* wurde zu 99,4 % und *Laufen* zu 100 % richtig erkannt. Lediglich *Gehen* hat mit 88,2 % eine geringere Genauigkeit, was durch die Ähnlichkeit mit *in Fahrzeug* zu erklären ist. Die Standardabweichungen überschneiden sich, da der Bereich von *Gehen* zwischen ca. 9 und 21 und der Bereich von *in Fahrzeug* zwischen 4 und 12 liegt. Dennoch ist die Standardabweichung ein sehr gutes Merkmal für die Bewegungserkennung.

Andere Untersuchungen verwendeten die Fourier-Transformation, um die Daten in den Frequenzbereich zu wandeln [21] [27] [16] [26]. Die Umwandlung in den Frequenzbereich ermöglicht eine weitere Sicht auf die Daten, aber benötigt Sensordaten, die in gleich großen Intervallen

vorliegen [27]. Die Fourier-Transformation an sich und die Aufbereitung der Daten sind daher mit großem Aufwand verbunden, der sich möglicherweise nicht lohnt, wenn über die berechneten Werte eine Bewegung nicht besser klassifiziert werden kann.

Da sich die beschriebenen zu berechneten Werte stets nur auf den Beschleunigungssensor beziehen, kann für die anderen Sensoren keine Aussage getroffen werden. Speziell für Rotationsvektor, Orientierung und das magnetische Feld ist dies schwierig, da sich die absoluten Werte je nach Bewegungsrichtung des Benutzers ändern. Lediglich das Gyroskop, die Schwerkraft und die lineare Beschleunigung sind dem Beschleunigungssensor ähnlich. Für die lineare Beschleunigung und Schwerkraft werden die Standardabweichung und der Mittelwert berechnet, jeweils für die drei Achsen und einmal von dem normierten Wert. Für das Gyroskop werden lediglich die Standardabweichung und der Mittelwert der normierten Werte berechnet.

Für das magnetische Feld wird nur die Standardabweichung der normierten Werte berechnet. Das hat den Vorteil, dass die absoluten Werte und somit sowohl Ausrichtung des Smartphones als auch die Richtung, in die der Benutzer am Gehen ist, keinen Einfluss auf diesen Wert haben. Die Normalisierung kann durchgeführt werden, da alle drei Achsen das magnetische Feld messen.

Anders verhält sich der Orientierungssensor, der in Kapitel 2.1.2 beschrieben wurde. Für die Zwecke dieser Arbeit werden lediglich Pitch und Roll verwendet, welche die Rotation um die x- bzw. y-Achse messen. Da Azimut einen Winkel zum magnetischen Norden der Erde misst, sind die Werte sehr stark von der Richtung abhängig, wie schon beim magnetischen Feld. Abbildung 8 soll verdeutlichen, dass in diesem Fall auch die Standardabweichung nicht aussagekräftig ist, da die Werte zu unterschiedlich schwanken. Durch die in diesem Fall kreisförmige Gehbewegung verändern sich sowohl die absoluten Werte der Messung als auch die Standardabweichung.

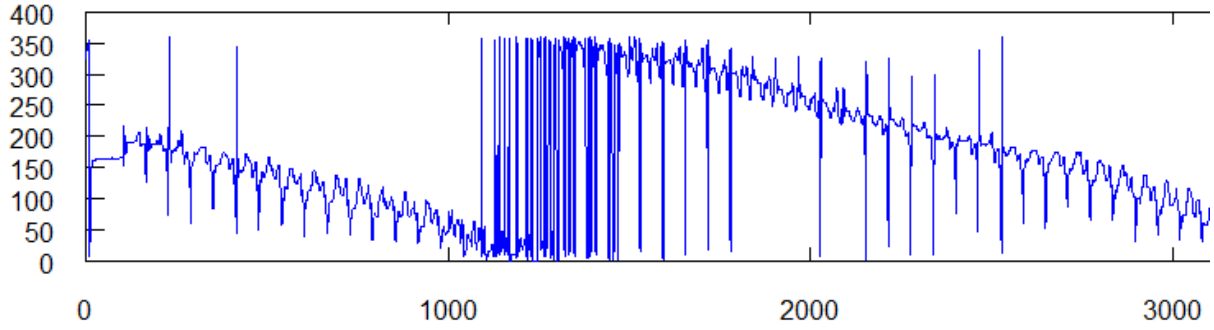

Abbildung 8 Azimut der Orientierung bei einer kreisförmigen Gehbewegung

Von Pitch und Roll werden jeweils die Standardabweichung berechnet und mit in die Trainingsphase der Klassifizierungsalgorithmen eingebunden.

Der Rotationsvektor weist ähnliche Merkmale auf wie der Azimut der Orientierung. Gemessen wird die Rotation des Gerätes um alle drei Achsen. Da das dazugehörige Koordinatensystem relativ zur Erde ist (Abbildung 4), sind die berechneten Werte ebenfalls abhängig von der Richtung der Bewegung des Benutzers. Da auch die Standardabweichung zu große Unterschiede während einer Bewegung aufweist, wird dieser Sensor für die Bewegungserkennung nicht verwendet.

4 Verwendete Softwarewerkzeuge (Tools)

Diese Bachelorarbeit wird durch verschiedene Open Source Tools unterstützt. Dazu gehören Tools für die Datensammlung und -verarbeitung. Im Folgenden werden diese Tools beschrieben.

4.1 Datensammlung

Für die Datensammlung wurden die zwei Applikationen *MotionProfile* und *Funf* verglichen. Da beide Applikationen für die Zwecke dieser Arbeit weniger geeignet waren, wurde eine eigene Applikation *DataRecording* geschrieben, die in Kapitel 4.1.3 beschrieben wird.

4.1.1 MotionProfile

MotionProfile entstand aus der Bachelorarbeit von Oliver Busch [4] und ist eine für das Betriebssystem Android 2.2 entwickelte Applikation, die Sensordaten des Smartphones sammelt.

Die Applikation bietet eine grafische Oberfläche (Abbildung 9) und ermöglicht dem User, verschiedene Sensordaten zu sammeln. Außerdem kann der Benutzer über die vier dargestellten Buttons seine Bewegungsart angeben, so dass zu jedem Datensatz die jeweilige Bewegung gespeichert wird.

Unterstütze Sensoren sind:

- Beschleunigungssensor
- Orientierung
- Magnetisches Feld (Kompass)
- GPS
- WLAN
- Bluetooth

Bei WLAN und Bluetooth wurden die erreichbaren Geräte gespeichert.

Der Benutzer kann vier verschiedene Bewegungen angeben:

- Gehen
- Laufen/Joggen
- Auto fahren
- Fahrradfahren

Abbildung 9 Aufzeichnung einer Bewegung mit *MotionProfile*

Die Applikation erwies sich für diese Untersuchung als weniger geeignet, da der Beschleunigungssensor, die Ausrichtung und der Kompass nur fünf Werte pro Sekunde aufgenommen haben und die Applikation den Schlafmodus des Smartphones nicht unterbindet, was zu einem Abbruch der Aufzeichnung führt, wenn das Display inaktiv wird. Weitere Sensoren fehlen, wie z.B. das Gyroskop, das erst seit Android Version 2.3 verfügbar war.

4.1.2 Funf

Funf ist ein *Open Sensing Framework* des MIT Media Lab, das kostenlos und Open Source verfügbar ist [40]. Das Funf Framework unterstützt eine große Anzahl an Sensoren, die für die Bewegungserkennung aber nicht alle benötigt werden. Der Vorteil gegenüber *MotionProfile* besteht darin, dass die angestrebten 50 Hz erreicht werden, der Schlafmodus verhindert wird und alle Sensoren vorhanden sind, die verwendet werden sollen. Abbildung 10 zeigt einen Ausschnitt von Funf, bei dem verschiedene Sensoren aktiviert werden können und die Aufzeichnung von Sensordaten über einen einfachen Button gestartet werden kann.

Funf stellt außerdem mit *funf in a box* [41] eine Möglichkeit bereit, mit der eine individuelle Applikation erstellt werden kann, bei der die zu erfassenden Daten vorher ausgewählt werden können. Die Daten werden nach jeder Aufzeichnung an den vorher verbundenen DropBox-Ordner gesendet, so dass das Smartphone nicht an den PC angeschlossen werden muss, um die Daten zu kopieren.

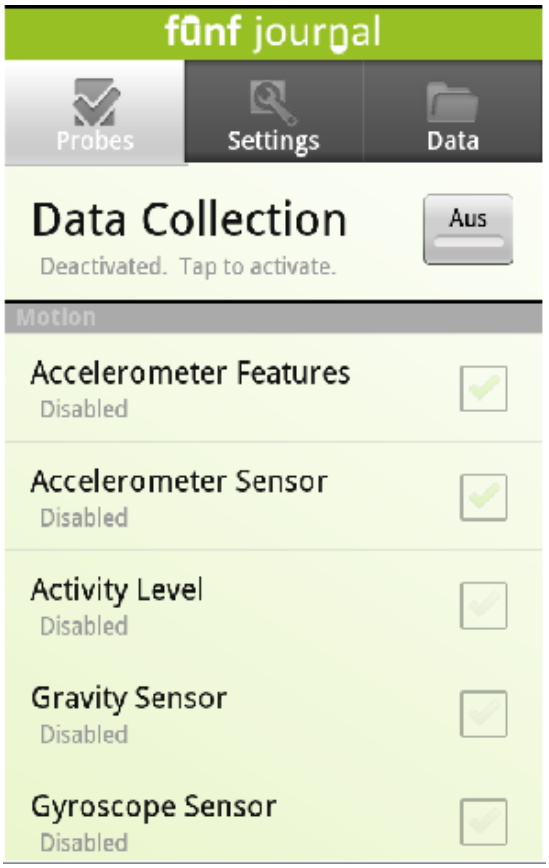

Abbildung 10 Ausschnitt der Sensorübersicht von *Funf*

Funf wurde nicht verwendet, da der Benutzer seine durchgeführte Bewegung nicht angeben kann, die für die Sammlung von Testdaten notwendig ist. Außerdem ist es nicht auf eine ununterbrochene Aufzeichnung von Sensordaten ausgelegt. Für jeden Sensor muss die Länge und die Periodizität einer Aufzeichnung angegeben werden. Es konnte zwar eingestellt werden, dass der Sensor nur einmal aufnehmen soll, aber die Länge der Aufzeichnung konnte nicht auf unbegrenzt gesetzt werden. Beim Testen wurden dazu teilweise keine Daten aufgenommen, so dass Funf auch nicht verwendet wurde.

4.1.3 DataRecording

Da die in Kapitel 4.1.1 und 4.1.2 dargestellten Datensammelapplikationen die Anforderungen für diese Bachelorarbeit nicht erfüllten, wurde eine eigene Applikation geschrieben, mit der sämtliche Daten mit ausreichender Abtastrate aufgenommen worden sind. Die wichtigsten Anforderungen sind dabei, dass die Bewegung der Aufzeichnung angegeben werden kann und dass die in Kapitel 3.2 ausgewählten Sensoren aufgenommen werden. *DataRecording* erfüllt diese Anforderungen und wurde für sämtliche Aufzeichnungen verwendet.

Der Benutzer kann verschiedene Angaben eingeben, die für die Bewegung relevant sind (Abbildung 11). Diese Angaben sind:

- Eine BenutzerID bzw. einen Eintrag, mit dem der Nutzer identifiziert werden kann (um verschiedenen Bewegungen einem einzelnen Benutzer zuordnen zu können)
- Die Ausrichtung des Smartphones
- Die Position des Smartphones
- Die Bewegung, die durchgeführt wird

Die Eingabe dieser Werte wird durch ein AutoComplete unterstützt (s. Abbildung 12), so dass die Angaben schnell erfasst werden können. Es können allerdings auch andere Ausrichtungen, Positionen oder Bewegungen als die vorgegeben angegeben werden.

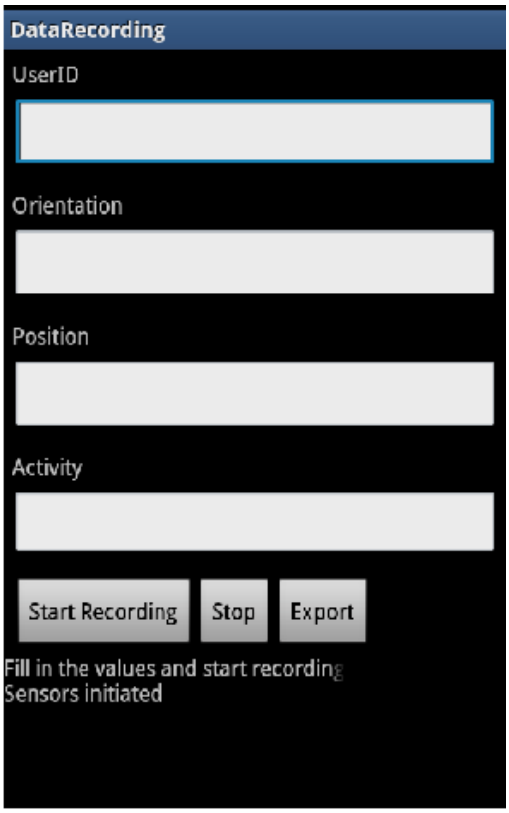

Abbildung 11 Screen von *DataRecording*

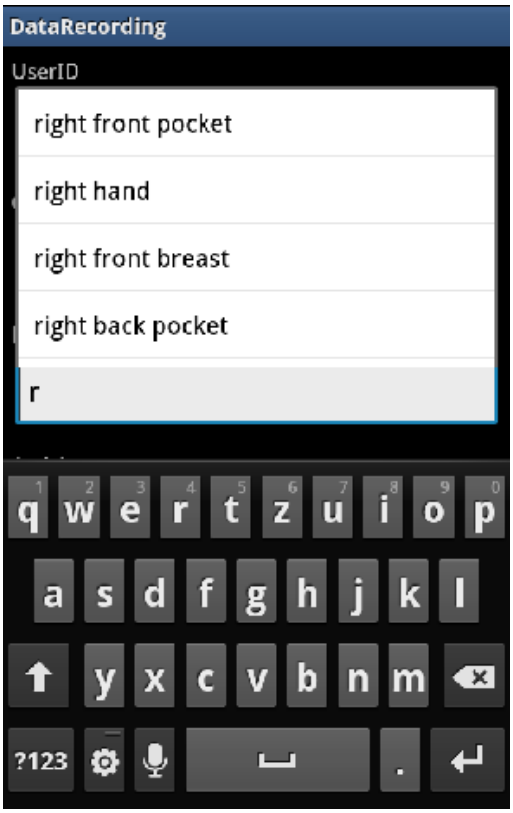

Abbildung 12 Screen von *DataRecording* mit AutoComplete

Abbildung 11 zeigt den Screen dieser Applikation. Der Button *Start Recording* startet die Aufnahme von Sensordaten, der Button *Stop* stoppt diese wieder und über den *Export*-Button werden die gesammelten Daten in .csv-Dateien geschrieben. Pro Sensor wird beim Exportieren eine eigene .csv-Datei auf dem Smartphone angelegt. Wenn die Bewegung gewechselt werden soll, müssen die bereits aufgenommenen Daten zuerst exportiert werden.

Ein wichtiger Unterschied zu den anderen Applikationen ist, dass die Sensordaten zur Laufzeit in Hashmaps gespeichert werden, für jeden Sensor wird eine Hashmap angelegt. Beim Exportieren der Daten entsteht daraus jeweils eine .csv-Datei pro Sensor, die in einem Ordner gespeichert werden. Der Vorteil der .csv-Dateien besteht in der einfachen Weiterverwendung in den folgenden Schritten. Im Ordner wird zusätzlich noch eine XML-Datei angelegt, in der die angegebenen Metadaten für die Bewegung stehen. Der Aufbau ist wie folgt:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Infos>
   <Orientation>Inhalt</Orientation>
   <Position> Inhalt </Position>
   <Activity> Inhalt </Activity>
</Infos>
```

Die XML-Datei enthält die Angaben zu Ausrichtung, Position und Bewegung, um bei der späteren Datenverarbeitung verschiedene Tests durchführen zu können. Am wichtigsten ist dabei die Angabe der Bewegung, da diese für die Trainingsphase der Klassifizierungsalgorithmen benötigt wird.

4.2 Datenverarbeitung mit RapidMiner

RapidMiner ist ein Open Source System für Data Mining [42]. Es unterstützt Klassifizierungsalgorithmen wie Decision Tree, Naive Bayes und Support Vector Machines, die für diese Arbeit verwendet werden. RapidMiner bietet eine grafische Oberfläche, über die Prozesse des Data Minings erstellt werden können. Jeder Prozess besteht aus einem oder mehreren Operatoren, die eine bestimmte Aufgabe erfüllen. Operatoren besitzen Ports, über die sie mit anderen Operatoren verbunden werden können. Man unterscheidet zwischen Input und Output Ports. Durch die Verknüpfung der Operatoren entsteht eine Kette, über die Daten fließen und verarbeitet werden.

Abbildung 13 zeigt einen Beispielprozess von RapidMiner. *Retrieve* und *Naive Bayes* sind Operatoren, die miteinander verbunden sind. Der Output Port *out* des *Retrieve* Operators ist mit dem Input Port *tra* des Operators Naive Bayes verbunden. Über diesen Port erhält der Operator *Naive Bayes* den Trainingsdatensatz, mit dem Naive Bayes angewendet wird.

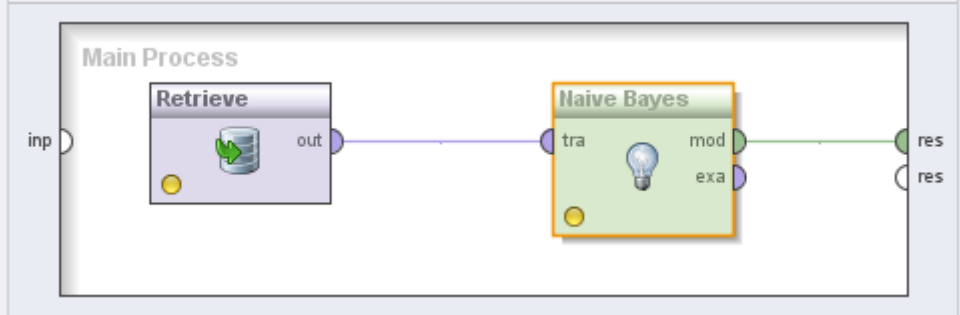

Abbildung 13 RapidMiner Beispielprozess

Die Operatoren der Klassifizierungsalgorithmen geben am Port *mod* ein Modell zurück, das als Ergebnis der Trainingsphase betrachtet werden kann und das die Repräsentation der Daten enthält. Da das Modell in der Android-Applikation verwendet werden soll, muss es über eine Schnittstelle exportiert werden können. Allerdings sind die Möglichkeiten begrenzt und es ist nur möglich, das Modell in einem speziellen XML-Format zu exportieren. Genaueres beschreibt Kapitel 5.4.

Mit RapidMiner kann das Modell nicht nur erstellt, visualisiert und gespeichert, sondern auch ausgewertet werden. Die Auswertung ist ein wichtiger Teil dieser Arbeit, um die Qualität des Ergebnisses zu erfassen. Die Ergebnisse werden dabei in Form einer sogenannten Confusion Matrix dargestellt, die eine Übersicht von erwarteter Bewegung gegen tatsächlich erkannte Bewegung bietet.

RapidMiner verfügt über viele Schnittstellen, die eine Arbeit mit verschiedenen Formaten ermöglicht. So können Daten z.B. über Datenbanken, Excel-Tabellen oder .csv-Dateien importiert werden. Außerdem können Prozesse im XML-Format gespeichert werden. Die für diese Arbeit erstellten Prozesse sind auf der beiliegenden CD im Ordner *RapidMiner* vorhanden.

5 Umsetzung

Nachdem die vorherigen drei Kapitel genauer erklärt haben, wie eine Bewegungserkennung theoretisch umgesetzt werden kann, beschreibt dieses Kapitel die praktische Umsetzung. Dazu gehört die Datensammlung, die mit der bereits beschriebenen Applikation *DataRecording* umgesetzt wird. Kapitel 5.1 beschreibt diese Applikation aus der technischen Sicht. Kapitel 5.2 befasst sich mit der Datenverarbeitung, die vor der Verwendung der Klassifizierungsalgorithmen nötig ist. Kapitel 5.3 beschreibt den Umgang mit RapidMiner bezüglich der vorher erfassten und verarbeiteten Daten und die Verwendung der Klassifizierungsalgorithmen. Als letztes wird in Kapitel 5.4 die Anwendung der Bewegungserkennung in einer Android Applikation beschrieben.

5.1 Datensammlung

Die Datensammlung wird mit der in Kapitel 4.1.3 beschriebenen Applikation durchgeführt. Die technischen Details der Applikation beschreibt dieses Kapitel. Abbildung 14 zeigt das UML-Diagramm von *DataRecording*. Es wurde auf die relevanten Aspekte reduziert.

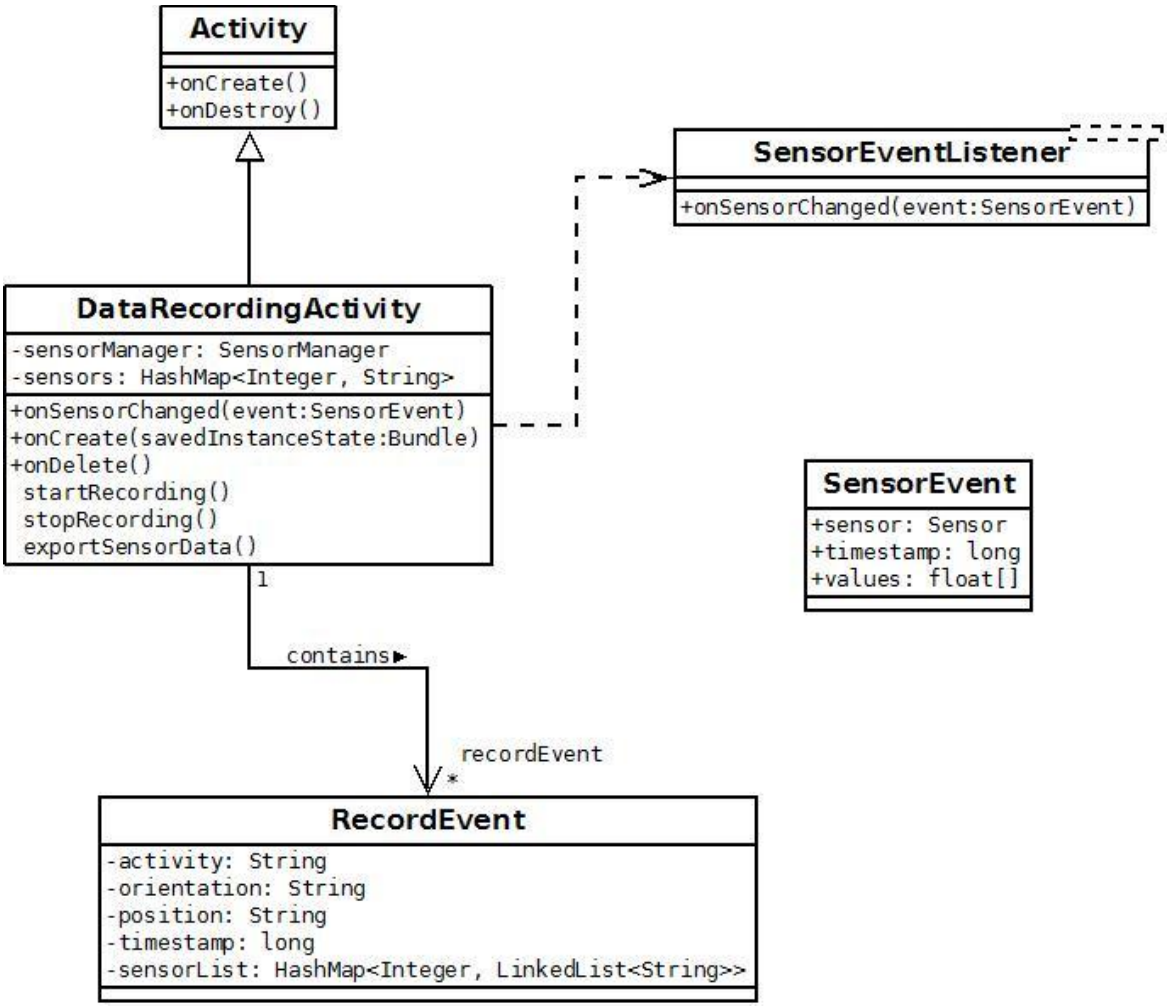

Abbildung 14 UML-Diagramm von *DataRecording*

Die Klasse *Activity* stammt aus der Android SDK und wird für eine Interaktion mit dem Benutzer verwendet [43]. In diesem Fall erbt die Klasse *DataRecordingActivity* von *Activity* und überschreibt

die zwei Methoden *onCreate()* und *onDestroy()*, die aufgerufen werden, wenn die Aktivität zum ersten Mal aufgerufen oder beendet und damit zerstört wird.

Das Interface *SensorEventListener* beinhaltet die Methode *onSensorChanged()* und ist für die Registrierung von Sensoren relevant. Sie wird von *DataRecordingActivity* implementiert und empfängt gelesene Sensordaten aller registrierten Sensoren. Die Registrierung der Sensoren wird über die Klasse *SensorManager* vorgenommen, die ebenfalls in der SDK vorhanden ist. Durch die Registrierung eines Sensors wird automatisch dessen Messung gestartet. Die verwendeten Sensoren werden in einer Hashmap in *DataRecordingActivity* gespeichert und der Klasse *RecordEvent* bei deren Instanziierung übergeben. Die Klasse *RecordEvent* repräsentiert eine Aufnahme einer Bewegung und enthält den eingegebenen Namen der Bewegung, die Ausrichtung und die Position des Smartphones. Dazu speichert das Attribut *timestamp* den Startzeitpunkt der Bewegung.

Da die Applikation die unverarbeiteten Sensordaten speichern soll und keine Berechnungen nötig sind, werden die ausgelesenen Sensordaten direkt in einer kommaseparierten Textdatei (csv) gespeichert. Die Methode *onSensorChanged()* wird für jede Sensormessung aufgerufen und enthält im Parameter *event* vom Typ *SensorEvent* die ausgelesenen Sensorwerte. Diese werden an das aktuelle *RecordEvent* übergeben und in folgender Reihenfolge gespeichert:

timestamp, erster Sensorwert, zweiter Sensorwert, dritter Sensorwert

Der erste bzw. zweite Sensorwert bezieht sich dabei auf den ersten bzw. zweiten Wert im Feld des Attributs *values* des *SensorEvents* und so weiter. Die verwendeten Sensoren enthalten alle jeweils nur drei Werte, die gespeichert werden. Lediglich der Rotationsvektor enthält einen vierten, optionalen Wert, der die Skalarkomponente des Rotationsvektors beinhaltet. Wichtig für die Weiterverarbeitung ist, dass der Timestamp die Zeit seit Start des Smartphones in Nanosekunden darstellt und nicht etwa die Millisekunden seit dem 01.01.1970.

Wenn die Aufnahme einer Bewegung gestoppt wird, werden auch die Messungen der Sensoren gestoppt. Daraufhin können die Daten exportiert werden. Die Exportfunktion liest alle in den Hashmaps gespeicherten Werte aus und schreibt für jeden Sensor eine .csv-Datei, wie bereits in Kapitel 4.1.3 beschrieben. Abbildung 15 zeigt einen Ausschnitt aus einer erstellten .csv-Datei für den Beschleunigungssensor.

	A	B	C	D
1	315085057977000	-0.7218784	5.87037	0.21792556
2	315085078930000	-1.607201	5.1484914	0.095342435
3	315085098359000	-2.3426998	4.739881	0.14982383
4	315085116969000	-2.4925237	4.69902	0.095342435
5	315085138958000	-2.3290794	4.344891	0.46309182
6	315085157234000	-1.8932283	4.276789	1.2803127
7	315085179751000	-1.3484144	5.47538	2.73769
8	315085198127000	-0.38136974	8.580819	5.1076303
9	315085218744000	1.5390993	14.001718	6.810174
10	315085240112000	2.4516625	16.712166	8.104107

Abbildung 15 Form der gespeicherten Sensordaten

Die in dieser Form von den verschiedenen Benutzern gespeicherten Daten müssen im nächsten Schritt so verarbeitet werden, dass sie von den Klassifizierungsalgorithmen verwendet werden können.

5.2 Datenverarbeitung

Für die Verarbeitung der gespeicherten Sensordaten müssen die verschiedenen .csv-Dateien der einzelnen Sensoren in eine einzige .csv-Datei umgewandelt werden. Jede Zeile dieser neuen Datei steht dabei für einen zwei Sekunden Ausschnitt der durchgeführten Bewegung und jede Spalte steht für einen berechneten Wert, wie z.B. den Mittelwert oder die Standardabweichung. Abbildung 16 zeigt einen Ausschnitt aus einer solchen .csv-Datei. Umgesetzt wird dieser Schritt mit einem selbst programmierten Skript, das im Java-Projekt *DataPreprocessing* enthalten und auf der CD vorhanden ist.

	A	B	C	D	E	F	G	H	I	J	
1	label	accS2Mean	accS2SD	accXMean	accYMean	accZMean	magS2SD	pitchSD	rollSD	linS2Mean	lin
2	Sitzen	9.889866	0.056340765	8.768381	2.0024745	4.110224	55.992554	0.3510938	124.91636	0.1421276	0.09
3	Sitzen	9.892779	0.03216593	8.803837	2.0034292	4.0424914	56.03551	0.1623839	125.73686	0.082485564	0.07
4	Sitzen	9.900248	0.025405258	8.811128	2.0039797	4.0448294	55.988754	0.12392648	125.752716	0.051633593	0.02
5	Sitzen	9.898115	0.026330788	8.806037	2.0041173	4.050745	56.025234	0.121350385	125.67056	0.04505174	0.02
6	Sitzen	9.889121	0.02660669	8.808238	1.9975138	4.0269446	55.98916	0.1561725	125.931526	0.05767904	0.03
7	Sitzen	9.887602	0.036573794	8.828071	1.9969089	3.9783916	56.00119	0.4255931	126.4812	0.10562153	0.06
8	Sitzen	9.885434	0.03832367	8.856281	2.001357	3.9073703	56.089745	0.42591375	127.26413	0.10793717	0.06
9	Sitzen	9.894229	0.037664838	8.883105	2.0012183	3.8698459	56.00287	0.21409605	127.75408	0.084590346	0.03
10	Sitzen	9.891956	0.038181458	8.8569765	1.9998282	3.9234936	55.94566	0.26657286	127.12817	0.094616055	0.04

Abbildung 16 Ausschnitt einer erstellten .csv-Datei

Die erstellte .csv-Datei wird im nächsten Schritt in RapidMiner benötigt, da sie als Input für die Klassifizierungsalgorithmen verwendet wird. Dafür muss die Datei in fünf Schritten importiert werden, die in Abbildung 17 bis Abbildung 21 gezeigt werden. Im ersten Schritt wird die Datei lediglich ausgewählt.

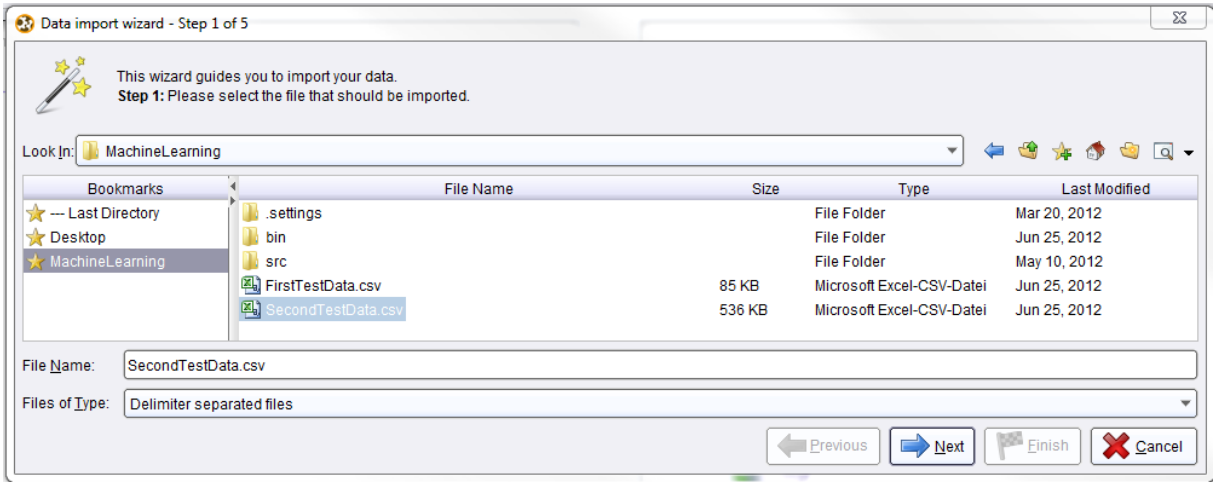

Abbildung 17 Schritt 1 des Imports

Wichtig im zweiten Schritt ist, dass die Spaltentrennung auf Komma gestellt wird, da ansonsten die komplette Zeile als eine Spalte interpretiert wird.

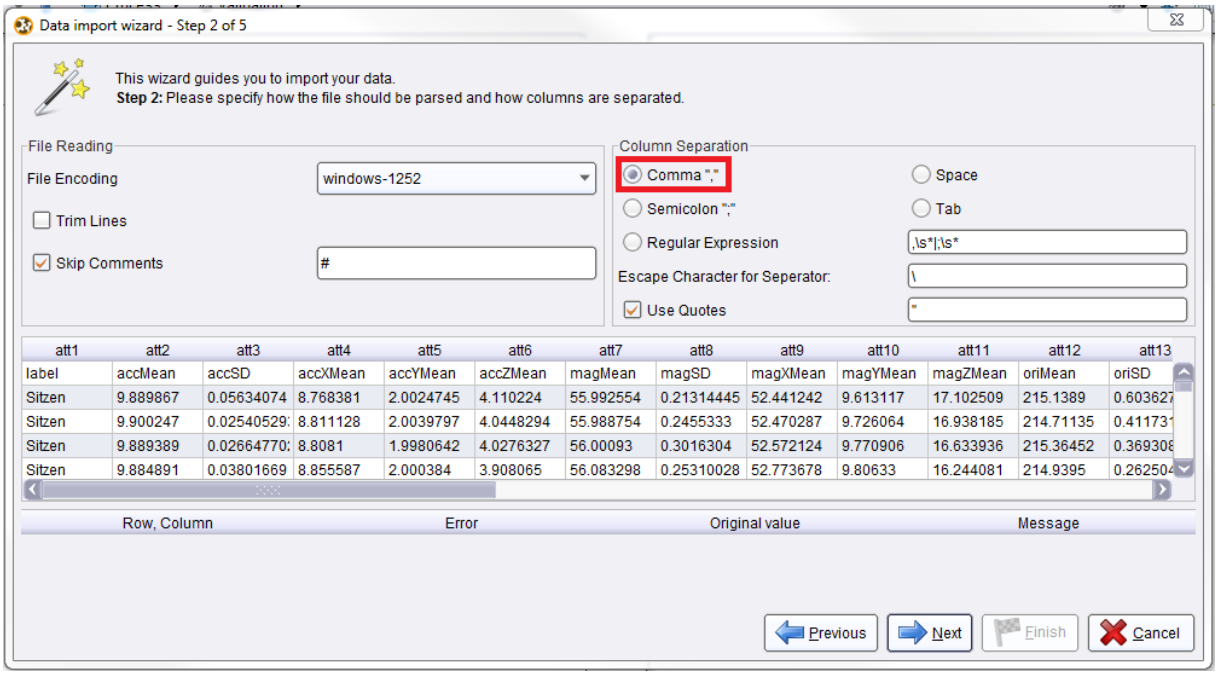

Abbildung 18 Schritt 2 des Imports

Da die erste Zeile Metadaten beinhaltet, welche die einzelnen Spalten benennen, muss in Schritt 3 die Zeile als *Name* markiert werden.

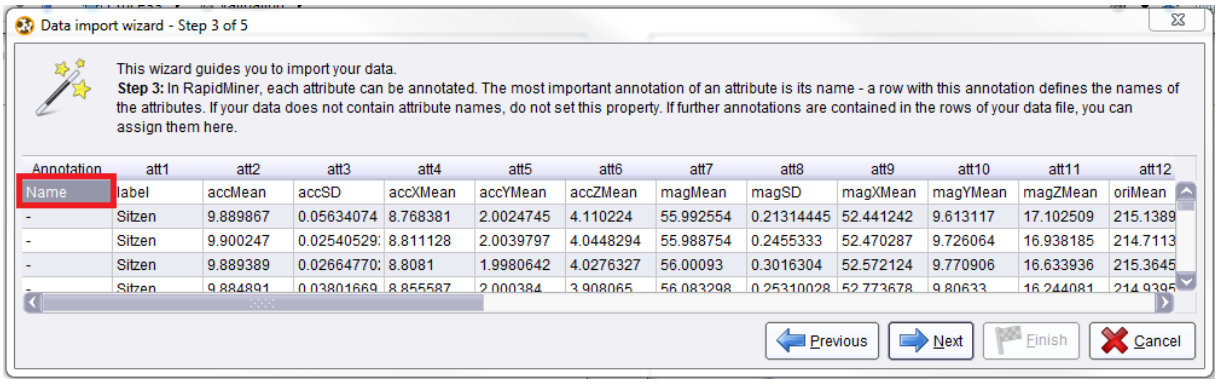

Abbildung 19 Schritt 3 des Imports

Auch in Schritt 4 müssen Änderungen vorgenommen werden. Da die erste Spalte das Label beinhaltet und dies der zu klassifizierende Wert ist, muss auch das speziell angegeben werden. Dazu kommt, dass die automatische Erkennung des Typs der Spalte nicht immer einwandfrei funktioniert und sichergestellt werden muss, dass der Typ auf *text* gestellt ist, da die automatische Erkennung es auf *binominal* setzt.

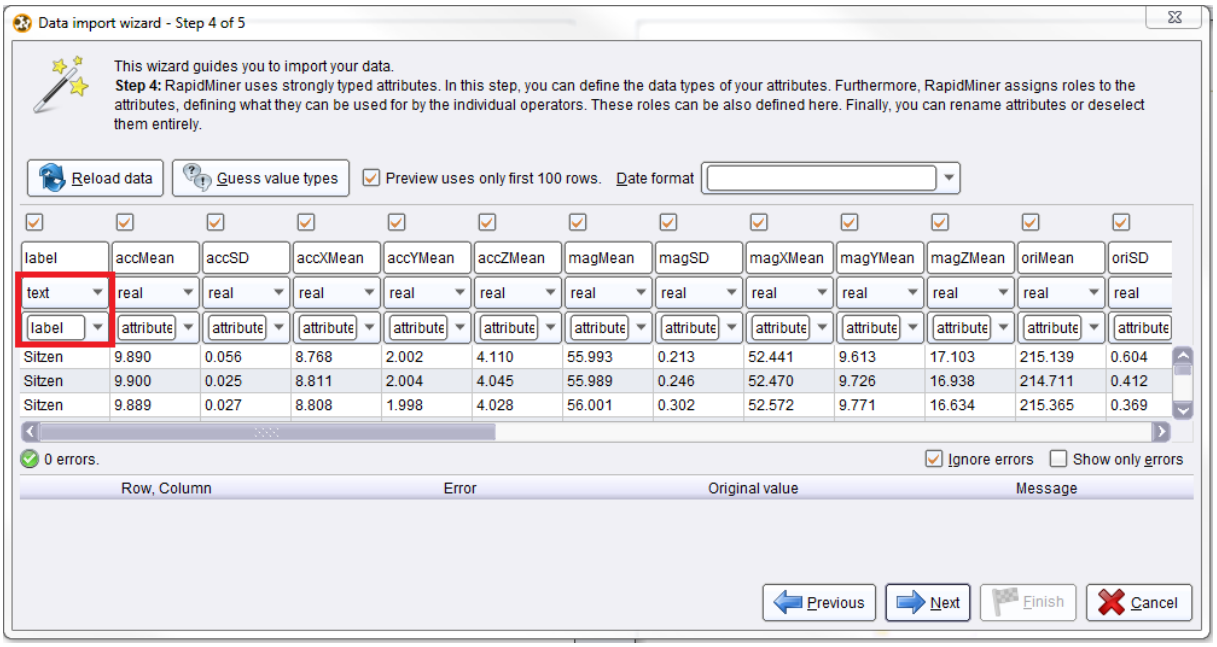

Abbildung 20 Schritt 4 des Imports

Im letzten Schritt muss lediglich der Speicherort und Dateiname angegeben werden.

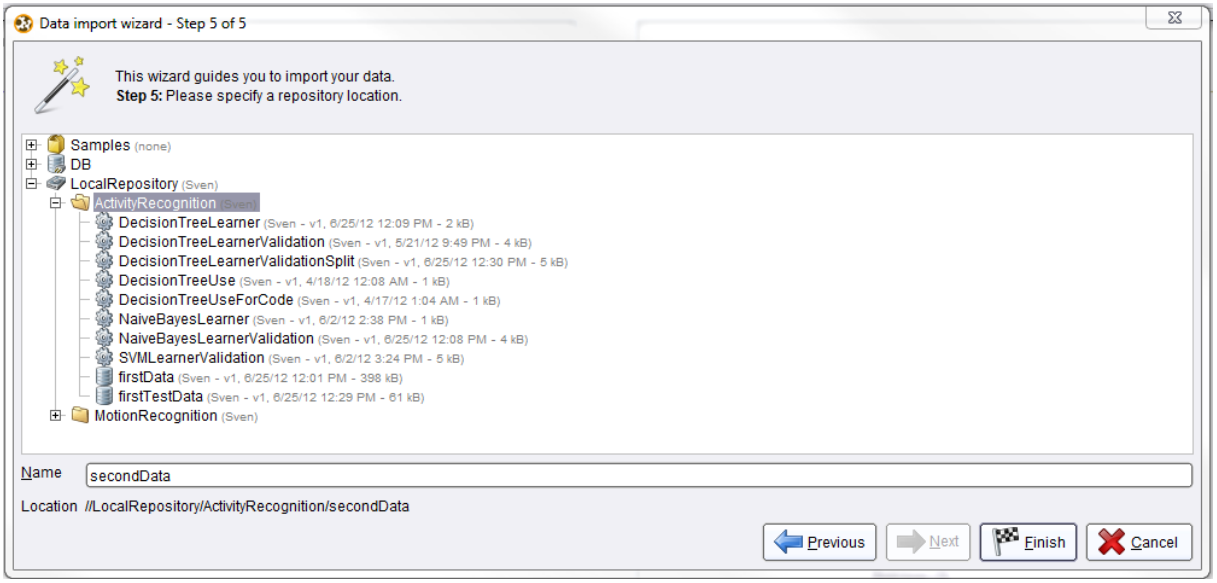

Abbildung 21 Schritt 5 des Imports

Nachdem die erstellte .csv-Datei importiert worden ist, kann sie in RapidMiner direkt verwendet werden. Dies behandelt das nächste Kapitel.

5.3 Data Mining

Mit den in Kapitel 5.2 importierten Daten können die weiteren Schritte mit der grafischen Oberfläche von RapidMiner durchgeführt werden. Dieses Kapitel befasst sich einerseits mit der Trainingsphase der Klassifizierungsalgorithmen und andererseits mit der Validierung der Ergebnisse.

Abbildung 22 zeigt den Prozess, mit dem ein Decision Tree erstellt werden kann. Der Operator *Retrieve* enthält den importierten Datensatz und ist mit dem Input Port *tra* des Operators *Decision Tree* verbunden, was der Port für den Trainingsdatensatz ist. Dadurch wird ein Modell eines Decision Tree auf Basis der übergebenen Daten erstellt. Der Operator *Write Model* schreibt eine XML-Datei, die das Modell des Decision Tree enthält und für den Validierungsprozess benötigt wird. Für Naive Bayes und SVM muss lediglich der *Decision Tree* Operator entsprechend ausgetauscht werden.

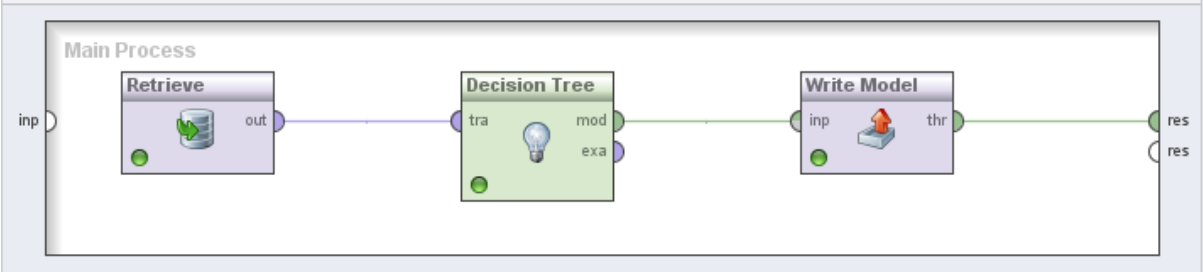

Abbildung 22 Trainingsphase des Decision Tree

Um die Genauigkeit der Klassifizierungsalgorithmen zu überprüfen, kann der Operator *Performance* verwendet werden. Abbildung 23 zeigt den gesamten Prozess der Überprüfung der Genauigkeit. *Read Model* erhält das aus dem vorherigen Prozess in XML-Format gespeicherte Modell des Decision Tree und *Retrieve* beinhaltet den Testdatensatz, der ebenfalls wie in Kapitel 5.2 importiert werden muss. Sowohl das Modell als auch der Testdatensatz werden dem *Apply Model* Operator übergeben, der den Testdatensatz gegen das übergebene Modell testet. Der *Performance* Operator wertet die Ergebnisse lediglich aus und präsentiert das Ergebnis in RapidMiner.

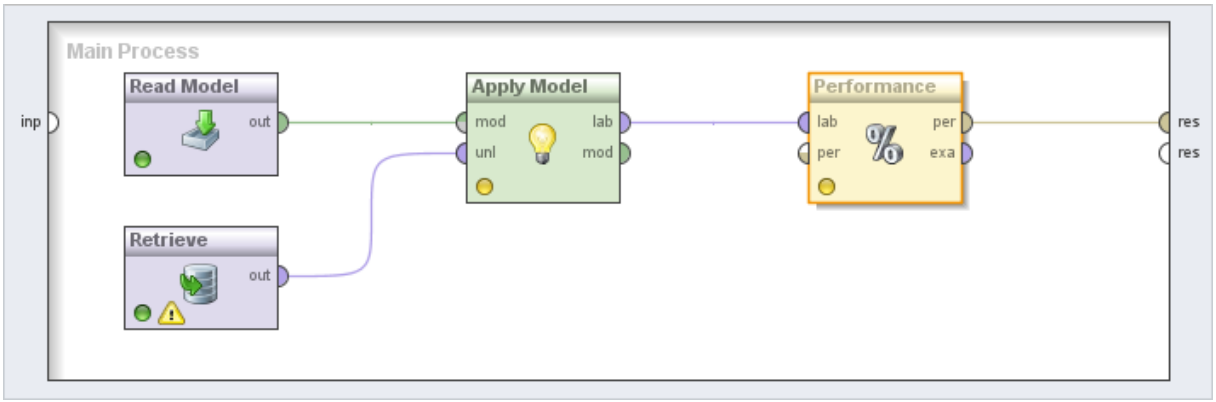

Abbildung 23 Überprüfung der Genauigkeit eines Modells

Bei der Verwendung des Modells ist die Bewegung vorher nicht bekannt. Da der Testdatensatz jedoch gelabelte Daten enthält, also die zu den Daten gehörige Bewegung bekannt ist, kann so das Modell überprüft werden. Die Daten des Testdatensatz werden mit Hilfe des Modells zugeordnet und anschließen mit der tatsächlichen Bewegung verglichen. Daraus wird eine so genannte Confusion Matrix erstellt, die einen Überblick über erkannte Bewegung gegenüber tatsächlicher Bewegung enthält. Die Ergebnisse werden in Kapitel 6 beschrieben und verwenden diese Form zur Präsentation der Daten.

5.4 Anwendung

Die erstellten Modelle werden von der Android-Applikation *ActivityRecognition* verwendet, die in diesem Kapitel vorgestellt wird. ActivityRecognition verwendet wie schon *DataRecording* Sensordaten und versucht mit Hilfe des vorher erstellten Modells die Bewegung zu klassifizieren. Die gemessenen Sensordaten müssen dabei wie in den vorherigen Verarbeitungsschritten behandelt werden. Das heißt, dass die Werte in Zeitfenster derselben Länge eingeteilt, dieselben statistischen Werte ermittelt und deswegen auch dieselben Sensoren verwendet werden müssen. Abbildung 25 zeigt das UML-Diagramm der Applikation *ActivityRecognition*. Die Klasse *Activity* und das Interface *SensorEventListener* haben dabei dieselbe Funktion wie schon in Kapitel 5.1 beschrieben.

Wichtig ist, dass die Klasse *ActivityRecognitionActivity* immer zwei Zeitfenster hat, in die Sensorwerte übergeben werden müssen. Die Klasse *TimeFrame* entspricht einem Zeitfenster. Jeder *TimeFrame* besitzt für jeden benötigten Sensor einen *SensorContainer*, in dem die Sensorwerte gespeichert und die statistischen Werte berechnet werden. Die Klasse *Statistics* muss dabei immer nur die im Modell benötigten Statistiken berechnet und den jeweiligen Zugriff über Getter Methoden bereitstellen. Nach Ablauf eines Zeitfensters wird die *analyzeTimeFrame()*-Methode der Klasse *ActivityRecognition* aufgerufen. Diese verwendet das erstellte Modell und klassifiziert das Zeitfenster. Das daraus resultierende Ergebnis wird mit Wahrscheinlichkeit in einem *AnalyzeResult* zurückgegeben und auf dem Bildschirm ausgegeben. Abbildung 24 zeigt den Screen von ActivityRecognition. Die Applikation liegt als .apk-Datei und als Source-Code im workspace auf der CD bei.

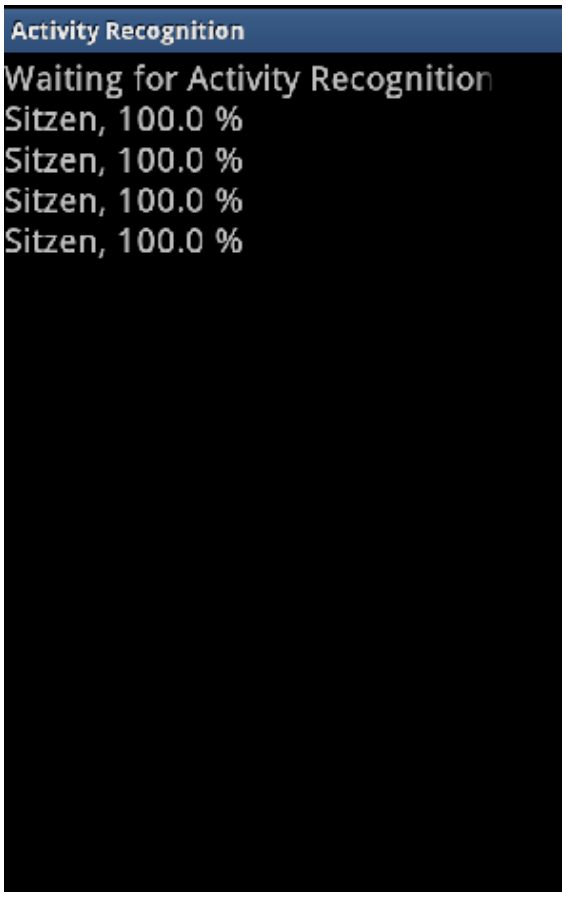

Abbildung 24 Screen der Applikation *ActivityRecognition*

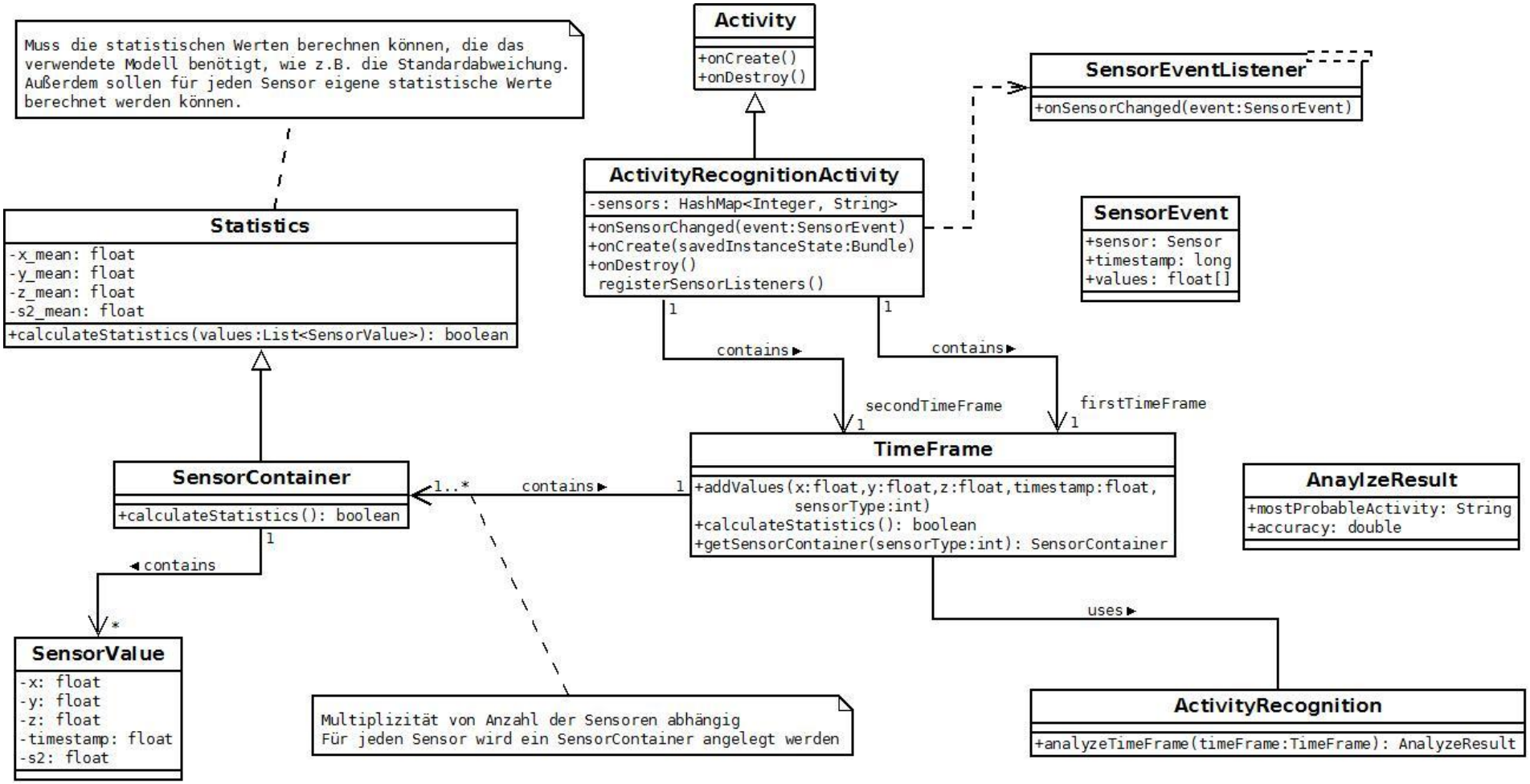

Abbildung 25 UML-Diagramm von *ActivityRecognition*

6 Ergebnis

Die erforderlichen Testdaten wurden mit dem Smartphone in der Hosentasche unter Verwendung der in Kapitel 4.1.3 beschriebenen Applikation gesammelt. Insgesamt sammelten sieben männliche Testpersonen im Alter von 21 bis 24 Jahren Daten für die Bewegungen *Sitzen*, *Stehen*, *Gehen* und *Laufen*. Die Aufnahme der Daten wurde unter Aufsicht durchgeführt, damit sichergestellt wurde, dass bei der Aufnahme keine andere als die angegebene Bewegung durchgeführt wird. Falsch gelabelte Daten führen zu einer geringeren Genauigkeit, auch wenn diese bei einer großen Anzahl an Daten nur gering ausfallen sollte.

Jede Testperson erhielt das gleiche Smartphone, um die Bewegungen durchzuführen. Verwendet wurde das Samsung Galaxy S2, das bei allen verwendeten Sensoren eine Abtastrate von 50 Hz unterstützt. Testweise wurde auch das Sony Ericsson XPERIA mini pro verwendet, das allerdings nur 25 Hz bei den Sensoren Magnetisches Feld und Orientierung erreicht und somit für das Experiment nicht zur Verfügung gestellt wurde.

Insgesamt wurden über 100 Minuten Testdaten gesammelt, wobei die vier Bewegungen nicht gleich häufig durchgeführt worden sind. Es wurden ungefähr 10 Minuten *Stehen*, 15 Minuten *Sitzen*, 35 Minuten *Gehen* und 42 Minuten *Laufen* aufgenommen. Da beim Sitzen und Stehen auch über einen längeren Zeitraum nur minimale Unterschiede der Sensormessungen zu erwarten sind, wurden nicht so viele Daten aufgezeichnet. Beim Laufen und Gehen hingegen können die Sensormessungen stärker schwanken, weshalb mehr Daten für diese Bewegungen gesammelt worden sind.

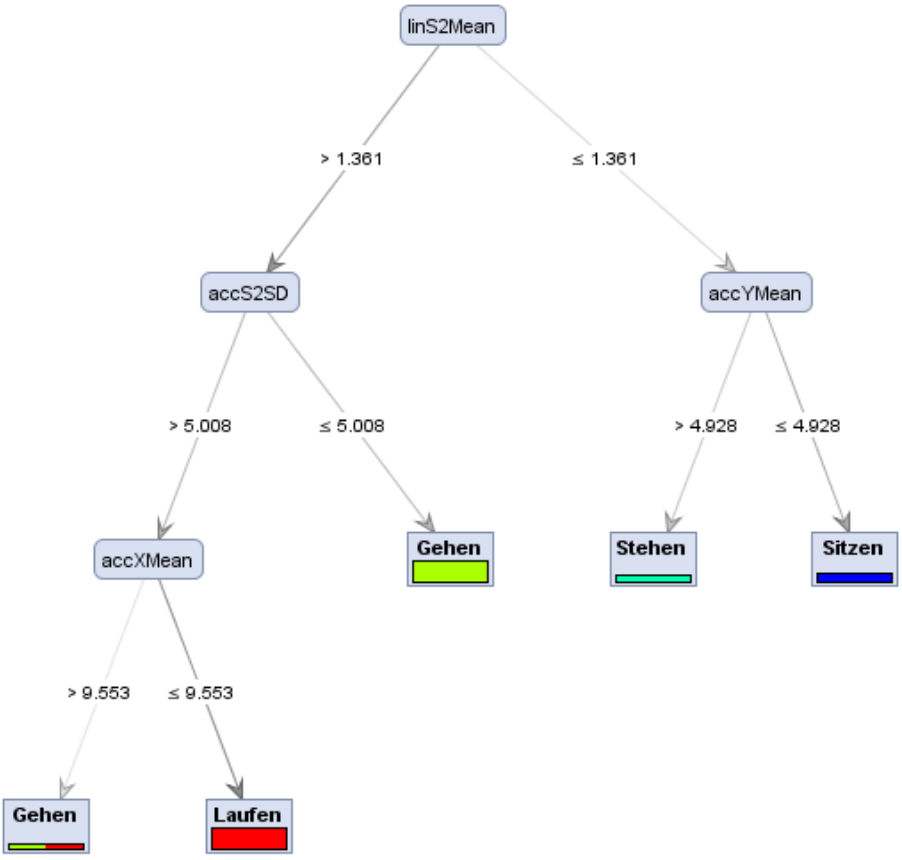

Abbildung 26 Erster erstellter Decision Tree

Diese Testdaten wurden für die folgenden Schritte so verarbeitet, wie es in Kapitel 5.2 und 5.3 beschrieben worden ist. Der erste Trainingsdatensatz verwendet Daten von sechs der sieben Testpersonen, während die Daten der siebten Person als Testdaten verwendet werden. Abbildung 26 zeigt den aus diesen Daten resultierenden Decision Tree. Bereits in diesem Modell gibt es eine geringe Überschneidung zwischen Gehen und Laufen. Diese befindet sich im Blatt des Baumes, das am weitesten links steht.

Der Trainingsdatensatz beinhaltet Daten vom Beschleunigungssensor, dem magnetischen Feld, der Orientierung und der linearen Beschleunigung. Die Schwerkraft konnte aufgrund fehlerhafter Werte der timestamps nicht verwendet werden. Für diesen Decision Tree wurden also 13 verschiedene Werte verwendet (vgl. Kapitel 3.7). Die Überprüfung der Genauigkeit erfolgt mit einem allgemeinen Testdatensatz, der wie bereits erwähnt aus den Daten einer Person erstellt wurde, deren Daten nicht im gezeigten Modell enthalten sind. Abbildung 27 zeigt die Confusion Matrix und damit die Genauigkeit des Decision Tree, der mit 98,69 % sehr gut abschneidet. Lediglich 12 Zeitfenster wurden als *Gehen* erkannt, die in Wahrheit *Stehen* waren.

accuracy: 98.69%

	true Gehen	true Laufen	true Sitzen	true Stehen	class precision
pred. Gehen	174	0	12	0	93.55%
pred. Laufen	0	312	0	0	100.00%
pred. Sitzen	0	0	203	0	100.00%
pred. Stehen	0	0	0	215	100.00%
class recall	100.00%	100.00%	94.42%	100.00%	

Abbildung 27 Confusion Matrix des ersten Decision Tree

Derselbe Trainingsdatensatz kann für Naive Bayes verwendet werden. Dabei wird eine Verteilungstabelle erstellt, die in Abbildung 28 dargestellt ist. Für jeden verwendeten Wert werden der Mittelwert und die Standardabweichung berechnet, so dass 26 Werte in der Tabelle aufgeführt werden. Anhand der Ähnlichkeit der Testdaten an diesen Trainingsdaten wird Naive Bayes die Bewegungen zuordnen. Ähnlich wie der eben erstellte Decision Tree könnte vor allem der Durchschnitt der Standardabweichung den Unterschied zwischen Gehen und Laufen ausmachen (Zeile 3 der Tabelle): 3,757 für Gehen und 6,125 für Laufen.

Die Überprüfung der Genauigkeit erfolgt mit demselben Testdatensatz wie schon bei dem Decision Tree. Abbildung 29 zeigt das Testergebnis in einer Confusion Matrix. Die Genauigkeit insgesamt ist mit 73,91 % im Gegensatz zum Decision Tree sehr gering. Der Grund ist die komplette Fehlklassifizierung der Bewegung *Stehen*, die aufgrund der Vorgehensweise von Naive Bayes auftritt. Die Wahrscheinlichkeit von *Stehen* ist aufgrund der Unterschiede zum Testdatensatz für die Klassifizierung für *Gehen* am höchsten.

Der dritte verwendete Klassifizierungsalgorithmus ist Support Vector Machine. Da wie in Kapitel 2.2.3 beschrieben SVM für ein Zweiklassenproblem ausgelegt ist, kann SVM in dieser Funktion nicht angewendet werden. Allerdings ist es möglich, mit RapidMiner dieses Problem zu lösen, da es einen Operator bereitstellt, der bei Verwendung mehrerer Klassen den SVM Algorithmus öfters hintereinander anwendet. Abbildung 30 zeigt die Confusion Matrix von SVM, die mit 53,06 % am schlechtesten abschneidet.

Attribute	Parameter	Sitzen	Stehen	Gehen	Laufen
accS2Mean	mean	9.920	9.822	10.862	12.658
accS2Mean	standard de	0.036	0.148	0.424	1.892
accS2SD	mean	0.036	0.036	3.757	6.125
accS2SD	standard de	0.019	0.027	0.505	0.537
accXMean	mean	6.987	4.140	2.924	3.254
accXMean	standard de	1.469	1.154	3.526	3.078
accYMean	mean	1.555	8.756	8.421	6.746
accYMean	standard de	0.479	0.765	2.558	1.675
accZMean	mean	6.418	0.848	0.668	1.102
accZMean	standard de	1.899	0.220	0.565	1.319
magS2SD	mean	49.940	48.080	44.121	48.758
magS2SD	standard de	4.602	1.176	7.884	3.481
pitchSD	mean	91.370	0.829	82.709	108.384
pitchSD	standard de	153.884	2.166	65.547	22.047
rollSD	mean	92.806	29.103	35.156	30.436
rollSD	standard de	26.342	34.097	40.662	8.287
linS2Mean	mean	0.066	0.128	23.736	24.125
linS2Mean	standard de	0.044	0.066	495.751	164.703
linS2SD	mean	0.037	0.073	23.945	19.760
linS2SD	standard de	0.047	0.047	552.993	195.645
linXMean	mean	0.004	0.009	2.707	5.793
linXMean	standard de	0.013	0.014	64.926	94.631
linYMean	mean	0.003	0.004	2.725	6.446
linYMean	standard de	0.007	0.004	58.716	87.241
linZMean	mean	0.005	0.015	6.311	4.733
linZMean	standard de	0.009	0.020	172.349	69.661

Abbildung 28 Naive Bayes Verteilungstabelle des ersten Trainingsdatensatz

accuracy: 73.91%

	true Gehen	true Laufen	true Sitzen	true Stehen	class precision
pred. Gehen	174	1	23	215	42.13%
pred. Laufen	0	311	0	0	100.00%
pred. Sitzen	0	0	192	0	100.00%
pred. Stehen	0	0	0	0	0.00%
class recall	100.00%	99.68%	89.30%	0.00%	

Abbildung 29 Confusion Matrix von Naive Bayes

accuracy: 53.06%

	true Gehen	true Laufen	true Sitzen	true Stehen	class precision
pred. Gehen	174	0	214	215	28.86%
pred. Laufen	0	312	1	0	99.68%
pred. Sitzen	0	0	0	0	0.00%
pred. Stehen	0	0	0	0	0.00%
class recall	100.00%	100.00%	0.00%	0.00%	

Abbildung 30 Confusion Matrix von SVM

Sowohl Naive Bayes als auch SVM sind sehr anfällig für Daten von Benutzern, deren Daten in der Trainingsphase nicht vorhanden waren. Sobald Trainingsdaten des Benutzers im Modell enthalten waren, lag die Genauigkeit bei über 99%. Einzig der Decision Tree konnte problemlos Bewegungen von Benutzern erkennen, deren Daten nicht in der Trainingsphase verwendet wurden. Daher werden die nächsten Tests auf Basis des Decision Tree erfolgen.

Smartphone Positionen

Der bisher verwendete Testdatensatz bestand aus Daten von Bewegungen, bei denen sich das Smartphone in gleicher Position und Ausrichtung befand. Zwei weitere Testdatensätze sollen die Robustheit des Decision Tree testen. Der erste Datensatz enthält die Bewegung *Gehen* mit sechs variierenden Positionen des Smartphones:

- Rechte und linke vordere Hosentasche
- Rechte und linke hintere Hosentasche
- In rechter und linker Hand

Um eventuelle Unterschiede zwischen den Positionen zu erkennen, werden die Positionen wie aufgelistet separiert betrachtet. Für die rechte und linke Hosentasche ergab der Testdatensatz eine Genauigkeit von 100 %. Für die rechte Hosentasche war dies annähernd zu erwarten, aber es zeigt, dass dies auch für die linke Hosentasche gilt. Auch für die beiden hinteren Hosentaschen wurde eine Genauigkeit von 100 % erreicht. Dies kann dadurch begründet werden, dass sich das Smartphone auch in der hinteren Hosentasche ausreichend bewegt, um nicht mit einer statischen Bewegung verwechselt zu werden. Überraschend ist, dass auch der Testdatensatz mit dem Smartphone in der linken oder rechten Hand eine Genauigkeit von 100 % aufweist. Dies kann aber an den Eigenschaften des Benutzers liegen, da die Hände beim Gehen immer in Schwingung waren.

Smartphone Ausrichtung

Der zweite Datensatz enthält die Bewegung *Gehen* mit vier variierenden Ausrichtungen des Smartphones:

- Display in Bewegungsrichtung zeigend und richtig herum
- Display in Bewegungsrichtung zeigend und falsch herum
- Display entgegen Bewegungsrichtung zeigend und richtig herum
- Display entgegen Bewegungsrichtung zeigend und falsch herum

Da die meisten Werte normiert wurden, ist eine hohe Genauigkeit zu erwarten, die mit 99,59 % auch erreicht wurde.

Einfluss der Sensoren

Wie in Abbildung 26 (S. 40) erkennbar ist, verwendet der Decision Tree lediglich Werte des Beschleunigungssensors und der linearen Beschleunigung. Die Auswahl dieser Werte hängt vom Algorithmus ab. Da immer die Werte ausgewählt werden, die Bewegungen am besten voneinander unterscheidet, kann keine Aussage über den Nutzen der anderen Sensoren getroffen werden. Daher werden die Trainingsdatensätze im Folgenden nur auf Basis eines einzigen Sensors erstellt. Somit entstehen fünf Datensätze für die Sensoren Beschleunigung, magnetisches Feld, Orientierung, lineare

Beschleunigung und Gyroskop. Zum Testen wird wieder der Datensatz verwendet, der aus Testdaten des Benutzers besteht, dessen Daten nicht als Trainingsdaten benutzt wurden.

Der nur auf Basis des Beschleunigungssensors erzeugte Decision Tree ähnelt stark dem ersten erstellen Decision Tree, aber verwendet nur noch Werte des Beschleunigungssensor und hat eine Genauigkeit von 100 %. Diese Erkenntnis bestätigt vergangene Untersuchungen, dass eine Bewegungserkennung unter ausschließlicher Verwendung des Beschleunigungssensors möglich ist.

Mit der linearen Beschleunigung ist die Unterscheidung zwischen *Sitzen* und *Stehen* sehr schwierig, da sich bei beiden Bewegungen das Smartphone nur unmerklich beschleunigt. Das führt zu einem sehr komplexen Decision Tree, da der Algorithmus versucht, eine Unterscheidungsmöglichkeit zu finden. *Laufen* und *Gehen* hingehen können anhand der Standardabweichung der normierten Werte sehr gut auseinander gehalten werden. Das verdeutlicht auch die in Abbildung 31 gezeigte Confusion Matrix. Während Laufen und Gehen zu 100 % erkannt wurden, kann zwischen Sitzen und Stehen kaum unterschieden werden. Die Genauigkeit bei diesen beiden Bewegungen ist dabei kaum besser als es beim einfachen Raten der Bewegung wäre. Die lineare Beschleunigung kann also zwischen den statischen Bewegungen nicht unterscheiden.

accuracy: 76.53%

	true Gehen	true Laufen	true Sitzen	true Stehen	class precision
pred. Gehen	174	0	9	0	95.08%
pred. Laufen	0	312	3	0	99.05%
pred. Sitzen	0	0	0	0	0.00%
pred. Stehen	0	0	203	215	51.44%
class recall	100.00%	100.00%	0.00%	100.00%	

Abbildung 31 Genauigkeit des Decision Tree der linearen Beschleunigung

Bei der Orientierung tritt dasselbe Problem auf. Zwar werden Gehen und Laufen zu fast 100 % klassifiziert, aber eine Unterscheidung zwischen Sitzen und Stehen ist nicht möglich.

Eine Unterscheidung mit der Standardabweichung der normierten Werte des magnetischen Feldes ist ebenfalls nicht möglich. Es werden alle Werte als Laufen klassifiziert und dementsprechend nur eine geringe Genauigkeit erreicht. Das liegt daran, dass die Standardabweichung bei allen Bewegungen sehr ähnlich war.

7 Fazit

Diese Arbeit untersuchte die Bewegungserkennung mit drei Klassifizierungsalgorithmen, vier Bewegungen, sieben Sensoren, vier Ausrichtungen und sechs Positionen des Smartphones. Wie vergangene Untersuchungen schon gezeigt haben, können ein Decision Tree, Naive Bayes und auch Support Vector Machines für die Klassifizierung eingesetzt werden. In dieser Untersuchung war der Decision Tree auch für Benutzer verwendbar, von denen keine Trainingsdaten für die Trainingsphase vorhanden waren. Für Naive Bayes und SVM galt dies nicht. Da in [11] diese Frage bereits erfolgreich getestet wurde, kann die geringe Genauigkeit in dieser Untersuchung entweder auf die geringe Anzahl an Testpersonen oder der Implementierung des Naive Bayes Algorithmus zugeordnet werden.

Eine Unterscheidung zwischen den Bewegungen *Gehen* und *Laufen* konnte mit den meisten Modellen problemlos umgesetzt werden. Die Unterscheidung zwischen *Sitzen* und *Stehen* hingegen konnte nur mit Hilfe des Beschleunigungssensors umgesetzt werden. Das liegt daran, dass der Beschleunigungssensor auch die Schwerkraft misst, die aufgrund der verschiedenen Ausrichtungen des Smartphones beim Sitzen und Stehen auf verschiedenen Achsen gemessen wird. Beim Stehen befand sich das Smartphone in einer vertikalen und beim Sitzen in einer horizontalen Position oder etwas schräg in der Hosentasche.

Die vier getesteten Ausrichtungen des Smartphones beeinflussten die Erkennung nicht und ebenfalls hatte keine der sechs getesteten Positionen des Smartphones einen Einfluss.

Wie bereits erwähnt ist der Beschleunigungssensor der einzige Sensor, durch den *Sitzen* und *Stehen* unterschieden werden konnten. Für die Bewegungen *Gehen* und *Laufen* jedoch konnten die Orientierung, lineare Beschleunigung und das Gyroskop eingesetzt werden. Da der Rotationsvektor womöglich keine charakteristischen Werte enthalten hätte, wurde dieser bereits vorher ausgeschlossen.

8 Zukünftige Arbeiten

Diese Arbeit hat bestätigt, dass die Bewegungserkennung mit einfachen Mitteln möglich ist. Bei der Betrachtung weiterer Bewegungen könnten mehr Informationen benötigt und somit neben Mittelwert und Standardabweichung weitere Werte berechnet werden. Speziell für das Treppensteigen wäre es interessant, ob die hier berechneten Werte ausreichen. Dazu kommen Bewegungen in Fahrzeugen, wie z.B. einem Bus, der Bahn, einem Auto aber auch auf dem Fahrrad. In diesen Fällen überschneiden sich Bewegungen, da der Benutzer beim Autofahren am Sitzen ist. Für diese Bewegungen kann auch GPS nützliche Informationen liefern.

Ein weiterer Punkt für zukünftige Arbeiten betrifft die Anzahl der Testpersonen. Da für diese Arbeit nur eine kleine Anzahl an Personen Testdaten gesammelt haben, ist das Ergebnis nicht sehr repräsentativ.

Durch die Entscheidung, dass die Bewegungserkennung zur Laufzeit durchgeführt wird, sind verschiedene Verbesserungsmöglichkeiten nicht umsetzbar gewesen. So wäre eine spätere Auswertung der Daten vorteilhaft, da z.B. der Kontext mit einbezogen werden könnte. Es kann versucht werden, Übergänge zwischen Bewegungen zu erkennen. Da Hinsetzen oder Aufstehen das Smartphone im Gegensatz zu nur *Sitzen* oder *Stehen* beschleunigt, könnte so besser auf diese Bewegungen geschlossen werden. Außerdem könnten sich überschneidende Bewegungen erkannt werden. Ein Benutzer, der die letzten fünf Minuten Auto gefahren ist, kann ohne eine Bewegung des Smartphones nicht woanders als im Auto sitzen. Sollte in solchen Fällen *Sitzen* ohne vorherige Bewegung erkannt werden, kann davon ausgegangen werden, dass der Benutzer immer noch im Auto sitzt und z.B. an einer Ampel oder im Stau steht. Ob dies dann als *Autofahren* oder *Sitzen* klassifiziert wird, kann über den Zweck der Applikation entschieden werden, welche die Bewegungserkennung einsetzt.

Anhang

Die mitgelieferte CD enthält die Daten, die für diese Bachelorarbeit relevant sind. Die folgenden Daten sind auf dieser CD zu finden:

root

Direkt auf der CD befindet sich eine *readme.txt*, welche die hier aufgeführten Informationen enthält.

Ordner *eclipse*

Enthält eine lauffähige Eclipse-Version mit einer installierten Android SDK, so dass die Android-Applikationen direkt ausgeführt werden können. Wichtig ist, dass der Emulator keine Sensordaten simuliert und somit die Applikationen nur mit angeschlossenem Smartphone funktionieren. Der Workspace muss möglicherweise angepasst werden. Die Projekte liegen im Ordner *workspace*.

Ordner *workspace*

Innerhalb dieses Ordners liegen die folgenden Projekte, deren Zweck an den jeweiligen Stellen dieser Arbeit erklärt wurde:

- ActivityRecognition (Android-Applikation)
- DataPreprocessing
- DataRecording (Android-Applikation)

Ordner *apk*

Enthält die kompilierten .apk-Dateien der DataRecording- und ActivityRecognition-Applikation. Diese können auf das Smartphone übertragen und ausgeführt werden. Damit die Applikationen installiert werden können, müssen möglicherweise *unbekannte Quellen* erlaubt werden. Dies ist eine Einstellung von Android, die verhindert, dass eine Installation von nicht-Market-Anwendungen zugelassen wird. Zu finden ist diese Option unter *Einstellungen* → *Anwendungen* unter Android 2.3.4.

Ordner *Sensordaten*

Enthält alle aufgenommenen Sensordaten der Testpersonen.

Ordner *RapidMiner*

Enthält die Prozesse von RapidMiner, die für diese Bachelorarbeit verwendet worden sind. Die Skripte liegen im XML-Format vor und können in RapidMiner direkt importiert werden. Die dazugehörigen Datensätze liegen im Unterordner *Datensätze* im .csv-Format.

Ordner *Dokument*

Enthält die digitale Version dieser Bachelorarbeit im .pdf-Format.

9 Literaturverzeichnis

[1] Y.-J. Hong, I.-J. Kim, S. C. Ahn und H.-G. Kim, „Mobile health monitoring system based on activity recognition using accelerometer," *Simulation Modelling Practice and Theory 18 (4),* pp. 446-455, 2010.

[2] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng und A. T. Campbell, „Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application," *SenSys '08 Proceedings of the 6th ACM conference on Embedded network sensor systems ,* pp. 337-350, 2008.

[3] A. Thiagarajan, J. Biagioni, T. Gerlich und J. Eriksson, „Cooperative transit tracking using smart-phones," *SenSys '10 Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems,* pp. 85-98, 2010.

[4] O. Busch, „Bereitstellen eines Frameworks für das Smartphone HTC Legend mit Android," Uni Koblenz-Landau, Koblenz, 2011.

[5] J. Fraden, Handbook of Modern Sensors - Physics, Desings, and Applications, London: Springer Science+Business Media, 2010.

[6] N. Györbíró, Á. Fábián und G. Hományi, „An activity recognition system for mobile phones," *Mobile Networks and Applications 14 (1),* Bd. 14, Nr. 1, pp. 82-91, 2009.

[7] Apple/Nike, „Apple," [Online]. Available: http://www.apple.com/ipod/nike. [Zugriff am 25 04 2012].

[8] Z. Prekopcsák, S. Soha, T. Henk und C. Gáspár-Papanek, „Activity Recognition for Personal Time Management," *AmI '09 Proceedings of the European Conference on Ambient Intelligence,* pp. 55-59, 2009.

[9] Y. Maruno, K. Cho, Y. Okamoto, H. Setoguchi und K. Ikeda, „An Online Human Activity Recognizer for Mobile Phones with Accelerometer," *ICONIP'11 Proceedings of the 18th international conference on Neural Information Processing - Volume Part II,* pp. 358-365, 2011.

[10] ActiServ, „Youtube, ActiServ," [Online]. Available: http://www.youtube.com/watch?v=7QMH29_y6yw. [Zugriff am 25 01 2012].

[11] T. S. Saponas, J. Lester, J. Froehlich, J. Fogarty und J. Landay, „iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones," 2008.

[12] A. M. Khan, Y.-K. Lee, S. Y. Lee und T.-S. Kim, „Human Activity Recognition via An Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis," *The 5th International Conference on Future Information Technology (FutureTech2010),* pp. 1-6, 2010.

[13] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke und M. Beigl, „ActiServ: Activity Recognition Service for mobile phones," *2010 International Symposium on Wearable Computers (ISWC),* pp. 1-8, 2010.

[14] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong und B. Krishnamachari, „A framework of energy efficient mobile sensing for automatic user state recognition," *MobiSys '09 Proceedings of the 7th international conference on Mobile systems, applications, and services,* pp. 179-192, 2009.

[15] A. Henpraserttae, S. Thiemjarus und M. Marukatat, „Accurate Activity Recognition using a Mobile Phone regardless of Device Orientation and Location," *2011 International Conference on Body Sensor Networks (BSN),* pp. 41 - 46, 2011.

[16] L. Sun, D. Zhang, B. Li, B. Guo und S. Li, „Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," *Proceedings of the 7th international conference on Ubiquitous intelligence and computing,* pp. 548-562, 2010.

[17] T. Brezmes, J.-L. Gorricho und J. Cotrina, „Activity Recognition from Accelerometer Data on a Mobile Phone," *IWANN '09 Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living,* pp. 796-799, 2009.

[18] J. Yang, H. Lu, Z. Liu und P. P. Boda, „Physical Activity Recognition with Mobile Phones: Challenges, Methods, and Applications," *Multimedia Interaction and Intelligent User Interfaces,* pp. 185-213, 2010.

[19] L. Grokop, A. Sarah, C. Brunner, V. Narayanan und S. Nanda, „Activity and device position recognition in mobile devices," *UbiComp '11 Proceedings of the 13th international conference on Ubiquitous computing,* pp. 591-592, 2011.

[20] A. M. Khan, „Human Activity Recognition Using A Single Tri-axial Accelerometer," Kyung Hee University, Seoul, 2011.

[21] S. Reddy, J. Burke, D. Estrin, M. Hansen und M. Srivastava, „Determining transportation mode on mobile phones," *ISWC '08 Proceedings of the 2008 12th IEEE International Symposium on Wearable Computers,* Nr. 25-28, 2008.

[22] J. R. Kwapisz, G. M. Weiss und S. A. Moore, „Activity Recognition using Cell Phone Accelerometer," *SIGKDD Explorations,* pp. 74-82, 2010.

[23] D. Sugimori, T. Iwamoto und M. Matsumoto, „A Study About Identification of Pedestrian by Using 3-axis Accelerometer," *17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications,* pp. 134-137, 2011.

[24] N. Ravi, N. Dandekar, P. Mysore und M. L. Littman, „Activity Recognition from Accelerometer Data," *IAAI'05 Proceedings of the 17th conference on Innovative applications of artificial*

intelligence - Volume 3, Nr. 3, pp. 1541-1546, 2005.

[25] J. Frank, S. Mannor und D. Precup, „Activity Recognition with Mobile Phones," *ECML PKDD'11 Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part III,* pp. 630-633, 2011.

[26] Z. Zhao, Y. Chen, J. Liu, Z. Shen und M. Liu, „Cross-People Mobile-Phone Based Activity Recognition," *IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume III,* pp. 2545-2550, 2011.

[27] Y.-C. Yang, T. Toida und C.-M. Hong, „Transportation prediction using build-in triaxial accelerometer in cell phone," 2010.

[28] Google, „Sensors | Android Developers," [Online]. Available: http://developer.android.com/guide/topics/sensors/index.html. [Zugriff am 05 03 2012].

[29] Google, „Sensors Overview | Android Developers," [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html. [Zugriff am 19 06 2012].

[30] Google, „Android 2.3 Compatibility Definition," 2010. [Online]. Available: http://static.googleusercontent.com/external_content/untrusted_dlcp/source.android.com/de//compatibility/android-2.3-cdd.pdf. [Zugriff am 20 6 2012].

[31] Google, „SensorEvent | Android Developers," 17 12 2011. [Online]. Available: http://developer.android.com/reference/android/hardware/SensorEvent.html#values.

[32] M. B. Kjærgaard, J. Langdal, T. Godsk und T. Toftkjær, „EnTracked: energy-efficient robust position tracking for mobile devices," *MobiSys '09 Proceedings of the 7th international conference on Mobile systems, applications, and services,* pp. 221-234, 2009.

[33] Heise Online, „Heise," [Online]. Available: http://www.heise.de/security/meldung/Beschleunigungssensor-spioniert-Smartphone-Tastatur-aus-1326115.html. [Zugriff am 25 01 2012].

[34] G. Paller, „Motion recognition with Android devices," 2011. [Online]. Available: http://www.slideshare.net/paller/motion-recognition-with-android-devices. [Zugriff am 2 7 2012].

[35] L. Rokach und O. Maimon, Data Mining with Decision Trees: Theory and Applications, Singapore: World Scientific Co. Ptw. Ltd., 2008.

[36] W. Konen, „http://www.gm.fh-koeln.de," [Online]. Available: http://www.gm.fh-koeln.de/~konen/WPF-DM-Cup/04-Naive-Bayes.pdf. [Zugriff am 19 07 2012].

[37] I. Rish, „An empirical study of the naive Bayes classifier," *IBM Research Report,* 02 11 2001.

[38] I. H. Witten und E. Frank, Data Mining - Practical Machine Learning Tools and Techniques, San Francisco: Elsevier Inc., 2005.

[39] T. Huynh und B. Schiele, „Analyzing Features for Activity Recognition," *sOc-EUSAI '05 Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies,* pp. 159-163, 2005.

[40] MIT Media Lab, „funf | Open Sensing Framework," 2011. [Online]. Available: http://funf.media.mit.edu/. [Zugriff am 11 12 2011].

[41] MIT Media Lab, „Funf in a Box," [Online]. Available: http://funf.media.mit.edu/inabox. [Zugriff am 22 06 2012].

[42] Rapid-I GmbH, [Online]. Available: http://rapid-i.com/. [Zugriff am 18 11 2011].

[43] Google, „Android Developers - Activity," [Online]. Available: http://developer.android.com/reference/android/app/Activity.html. [Zugriff am 16 07 2012].

10 Abbildungsverzeichnis

11 Tabellenverzeichnis