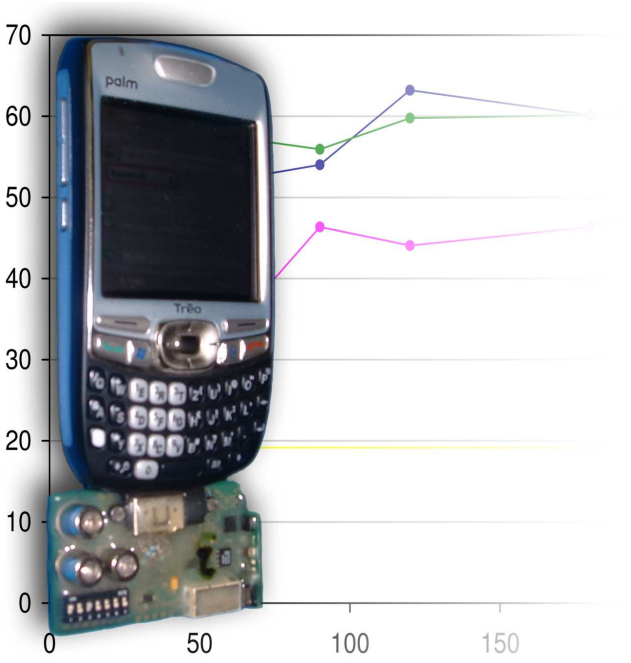
Universität Zürich
Institut für Informatik

Dynamic and Distributed
Information Systems

Voraussage von Benutzerverhalten in dynamischen Umgebungen



Diplomarbeit 4. Juni 2007

Domenic Benz

of Raperswilen TG, Switzerland

Student-ID: 02-722-064

d.benz@access.unizh.ch

Betreuer: **Peter Vorburger**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich

http://www.ifi.unizh.ch/ddis

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, welche mich während der Durchführung dieser Diplomarbeit unterstützt haben.

Herzlichen Dank an Prof. Bernstein und Peter Vorburger, welche mir die Möglichkeit gaben, meine Diplomarbeit in diesem interessanten Themengebiet zu absolvieren. Peter Vorburger möchte ich für die hervorragende Betreuung und Unterstützung mit Rat und Tat während der gesamten Arbeit danken.

Ein besonderer Dank gebührt abschliessend meiner Freundin und meiner Familie für die persönliche Unterstützung während meines Studiums und in der Zeit dieser Diplomarbeit.

Zusammenfassung

Die zunehmende Verbreitung von Mobiltelefonen hat einen erheblichen Einfluss auf den modernen Alltag. Sie bringt neben vielen Vorteilen jedoch auch den negativen Effekt unerwünschter Störungen und Unterbrüche mit sich. Vielfach wäre es wünschenswert, wenn sich ein Mobiltelefon der aktuellen Situation anpassen würde. Damit eine solche Anpassung möglich wird, muss das Mobiltelefon jedoch über Informationen zum aktuellen Kontext verfügen.

Zu diesem Zweck wurde in der vorliegenden Arbeit eine Software entwickelt, welche Daten aus einer Vielzahl von Sensoren erfasst. Die entwickelte Software wird in einem ersten Experiment eingesetzt. Dabei wird versucht, herauszufinden, ob sich aufgrund der Sensordaten, welche durch das mit der Software und diversen Sensoren ausgerüstete Mobilephone erfasst werden, Aussagen über die Aktivität und den Aufenthaltsort des Benutzers machen lassen. Die im Rahmen dieser Arbeit entwickelte Software sowie das durchgeführte Experiment sollen als Basis für weitere Forschung im Bereich der mobilen Kontexterfassung und der Unterbrechbarkeitsanalyse dienen.

Abstract

The increasing proliferation of mobile phones has a significant influence on our daily lifes. Allthough the increasing use of mobile devices has brought several advantages, it also has the negative effect of unwanted disturbance and interruptions. It is desirable that a mobile phone has the ability to adapt to the current situation it is in. For such an adaption to become possible, the mobile phone would need to have information about its current context.

To achieve this goal, a software is implemented which gathers data from a variety of sensors on a mobile phone. This software is then being used in a prototype experiment. In this experiment we try to determine if it is possible to predict a users activity and location based on the collected data.

The software implemented in this thesis and the results of the experiment help to prepare and conduct follow-up experiments in the field of context awareness and human interuptibility research.

Inhaltsverzeichnis

1

Einleitung

Mobiltelefone sind aus dem modernen Alltag kaum mehr wegzudenken. Sie erleichtern die zwischenmenschliche Kommunikation, sind Arbeitsgeräte oder dienen auch nur zur Unterhaltung. Die durch permanente Erreichbarkeit enorm verkürzten Kommunikationswege bringen viele Vorteile. So werden Entscheidungen schneller gefasst und auch kommuniziert oder auch Kontakte gepflegt, welche ansonsten vergessen gingen. Ebendiese permanente Erreichbarkeit sorgt jedoch auch für negative Effekte, wie zum Beispiel Stress durch zu viele Telefonate, oder auch drastische Verschlechterung der Produktivität durch Anrufe zum falschen Zeitpunkt. Da ein Anrufer normalerweise keine Informationen darüber hat, welcher Aktivität der Angerufene gerade nachgeht und an welchem Ort er sich befindet, muss die Entscheidung, welche Anrufe in der aktuellen Situation störend und welche wichtig sind vom Angerufenen getroffen werden. Bereits diese Entscheidung kann jedoch schon negativen Einfluss auf die Tätigkeit oder auch das Umfeld des Angerufenen haben.
An diesem Punkt möchte diese Arbeit ansetzen und erste Erkenntnisse generieren, welche allenfalls zukünftig helfen könnten, unerwünschte Störungen und Stress, verursacht durch Anrufe auf Mobiltelefone zu vermeiden.

1.1 Ziel dieser Arbeit

Das Ziel der vorliegenden Arbeit besteht darin, zu evaluieren ob aufgrund von Umgebungsdaten eines Mobiltelefonbenutzers Aussagen über seinen Kontext machbar sind (Definition Kontext: [Dey and Abowd, 1999]). Es soll dabei insbesondere untersucht werden, ob aufgrund früherer Daten Rückschlüsse auf die Aktivität und den Aufenthaltsort des Benutzers gemacht werden

können. Das Fernziel, welches jedoch nicht Thema dieser Arbeit ist, besteht darin, herauszufinden ob aufgrund des Kontexts Vorhersagen über den Grad der Störung durch einen Telefonanruf respektive über die Unterbrechbarkeit des Benutzers - welche sehr eng mit der Aktivität und dem Aufenthaltsort gekoppelt ist - machbar sind. Zu diesem Zweck wird eine Software entwickelt, welche direkt auf dem Mobiltelefon des Benutzers ausgeführt wird. Diese Software sammelt dabei kontinuierlich Daten aus der Umgebung des Benutzers.
Die Auswertung dieser Daten soll danach Rückschlüsse über den Kontext des Benutzers erlauben.

1.2 Verwandte Arbeiten

Mit der zunehmenden Verbreitung von Smart Devices in unseren täglichen Gebrauch sind verschiedene Arbeiten, Forschungen und Experimente durchgeführt worden, mit dem Ziel, diese Geräte - hauptsächlich durch Context Awareness - wirklich intelligent zu machen. Beispielsweise wurde in [Gellersen et al., 2002] das TEA (Technology Enabling Awareness) Projekt vorgestellt, welches anhand von verschiedenen Sensoren, angebracht auf einem separaten Modul und verbunden mit einem Mobilephone, den Kontext eines Benutzers erkennt. Neben der Arbeit von Gellersen et al. wurde auch in [Farringdon et al., 1999] untersucht, inwiefern sich die Aktivität einer Person aufgrund von Sensordaten aus einem Accelerometer erkennen lässt. Die Sensoren wurden jedoch in dieser Arbeit direkt am Körper (Gurt respektive Jacke) der Testperson getragen. Dies erlaubt eine bessere Nutzung der Daten, da die Ausrichtung relativ zum Träger immer gleich bleibt, während sie sich beispielsweise bei einem mobilen Gerät ständig ändern kann.

Weiter wurden verschiedene Arbeiten durchgeführt, welche die Unterbrechbarkeit einer Person mittels verschiedenen Sensordaten untersuchen. Arbeiten in diesem Bereich wurden beispielsweise vorgestellt in [Hudson et al., 2003], [Whitaker and Kay, 2005] & [Mühlenbrock et al., 2004]. In diesen Arbeiten wurden verschiedene, in einem Raum installierte Sensoren verwendet, um Voraussagen über die Unterbrechbarkeit der Person im jeweiligen Raum zu machen. Dies wurde hauptsächlich durch das Erkennen der Aktivität mittels Sensoren wie Video und Audio möglich.

1.3 Aufbau der Arbeit

Um die im Kapitel 1.1 beschriebenen Ziele zu erreichen sind verschiedene Schritte notwendig. Die vorliegende Arbeit ist demzufolge in drei Hauptteile gegliedert. Im ersten und zweiten Teil werden die technischen Randbedingungen sowie die Implementation der Software, welche für spätere Experimente benötigt wird beschrieben. Im dritten Teil dieser Arbeit wird das eigentliche Experiment zum Kontext eines Mobiltelefonbenutzers durchgeführt und beschrieben.

2

Technologien

Für die Durchführung dieser Arbeit sind einige technische Komponenten von zentraler Bedeutung. Aus diesem Grund sollen in diesem Abschnitt die technischen Rahmenbedingungen erläutert werden.
Dieses Kapitel befasst sich folglich zuerst mit der verwendeten Hardware, deren Besonderheiten sowie den daraus folgenden Konsequenzen für diese Arbeit. Danach soll kurz auf die zur Implementation verwendete Software eingegangen werden. Dies umfasst sowohl die zur Entwicklung notwendige Software als auch verwendete Frameworks von Drittherstellern sowie unabhängige Software zur Durchführung und Auswertung des später beschriebenen Experiments.

2.1 Hardware

An dieser Stelle sei nun auf die zum Einsatz kommende Hardware eingegangen. Insbesondere sollen dabei die spezifischen Vor- und Nachteile sowie die Besonderheiten der einzelnen Komponenten mit Bezug auf ihre Auswirkungen auf diese Arbeit aufgezeigt werden.

2.1.1 Palm Treo 750v

Das Zielgerät für die Implementation der Software und die später folgenden Exerimente ist der Treo 750v (Abbildung 2.1) der Firma PalmOne (Palm, Inc.) [1].

Da das unter Kapitel 2.1.2 beschriebene Sensorboard ursprünglich für den Palm Treo 650 entwickelt wurde und bereits erste Tests mit diesen Geräten durchgeführt wurden ist der Treo 750v

[1] `http://www.palm.com`, Abrufdatum: 25.04.2007

Abbildung 2.1: Palm Treo 750v

als Nachfolgemodell für diese Arbeit die geeignete Plattform. Ein weiteres entscheidendes Kriterium für die Wahl des Tro 750v bestand darin, dass dieses Gerät über den selben MultiConnector Anschluss wie das Vorgängermodell verfügt. Dieser Anschluss wird durch das in dieser Arbeit verwendete Serialboard (siehe Abschnitt 2.1.2) verwendet.

Dieses Gerät ist eines der ersten Windows Mobile basierten Geräte der Firma PalmOne (Windows Mobile Version 5.0). Die Verwendung von Windows Mobile als Betriebssystem bietet verschiedene Vorteile im Vergleich zur Verwendung des normalerweise auf Geräten der Firma PalmOne installierten PalmOS. Der für diese Arbeit mit Sicherheit wichtigste Vorteil ist es, dass Windows Mobile im Gegensatz zu PalmOS echtes Multithreading unterstützt.

Zur Benutzerinteraktion bietet der Treo 750v zwei Alternativen. Zum einen kann das Gerät mittels Stift oder Finger via Touchscreen bedient werden. Zum anderen steht eine vollwertige QWERTZ Tastatur zur Verfügung. Die im Rahmen dieser Diplomarbeit entwickelte Software wird primär für die Bedienung mittels Touchscreen entwickelt. Weiterführende Informationen und technische Spezifikationen zum Palm Treo 750v finden sich auf der angegebenen Internetseite [2].

[2] `http://www.palm.com/us/products/smartphones/treo750/specs.html` , Abrufdatum: 25.04.2007

2.1.2 Sensorboard

Eine zentrale Komponente dieser Arbeit ist das von Peter Vorburger [3] am Institut für Informatik (IFI) der Universität Zürich entwickelte Sensorboard (Abbildung 2.2).

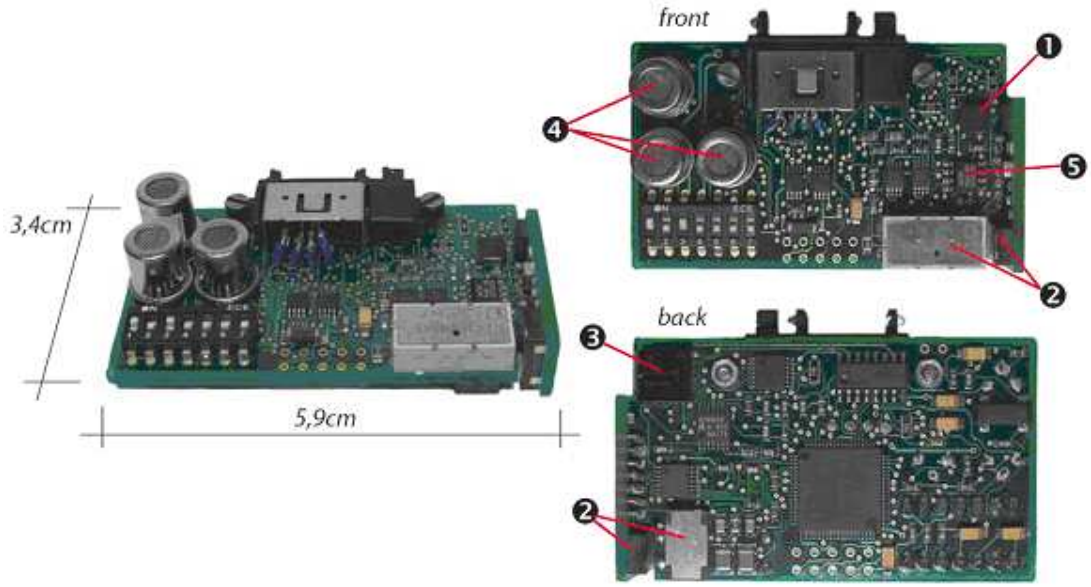


Abbildung 2.2: Sensorboard

Dieses Sensorboard besteht aus einer Reihe von Sensoren welche auf einer Platine angebracht sind und von einem Prozessor gesteuert werden. Zur Verfügung für diese Arbeit stehen somit folgende Messwerte:

- Umgebungstemperatur (Nr. 5)
- Beschleunigung (Nr. 1)
- Winkelbeschleunigung (Nr. 2)
- Verschiedene Gaswerte (Nr. 4)
- Magnetfelddaten (Nr. 3)

[3] `http://www.ifi.unizh.ch/ddis/people/vorburger/` , Abrufdatum: 25.04.2007

Der auf der Platine angebrachte Prozessor verfügt über einen 12-bit Analog Digital Wandler (ADC). Dieser ADC konvertiert die von den jeweiligen Sensoren gelieferten analogen Spannungswerte (0.5 - 2.5V) in digitale Messwerte zwischen 0 und 4096 (2^{12}).

Sensordatenformat

Die Sensordaten werden vom Sensorbaord in Form einer Komma separierten Liste zum Gerät übertragen. Jede übertragene Linie besteht dabei aus 12 hexadezimalen Werten, welche je einen Sensor repräsentieren. Listing 2.1 zeigt einen Ausschnitt aus einem solchen, vom Sensorboard übertragenen Datenstrom.

```
91C,79D,7AE,806,85F,7FC,7FD,2D3,2D0,8CF,856,70E
914,791,799,80B,85F,806,806,2D3,2D2,8E4,851,702
908,798,795,808,85F,804,806,2CE,2D0,8E1,854,70F
918,793,796,803,863,7FC,7FF,2D3,2CF,8E5,83E,713
911,789,7A2,803,861,803,806,2D6,2D5,8DD,84F,708
```

Tabelle 2.1: Format der Sensordaten

Die Werte in einer Zeile entsprechen dabei wie folgt den einzelnen Sensoren (siehe Listing 2.1, von links nach rechts):

1. Temperatur
2. Accelerometer y-Achse
3. Accelerometer x-Achse
4. Accelerometer z-Achse
5. Gyroskop y-Achse
6. Gyroskop x-Achse
7. Gyroskop z-Achse
8. Gas Sensor 1 (CO)
9. Gas Sensor 2/3 (Brennbare Gase?/Alkohol?)
10. Magnetfeldsensor x-Achse

11. Magnetfeldsensor z-Achse
12. Magnetfeldsensor y-Achse

2.1.3 GPS

Um die geographische Position als zusätzlichen Messwert zur Verfügung zu haben, kam ein GPS Empfänger zum Einsatz. Es handelt sich dabei um den GPS Empfänger Altina GBT-709 (Abbildung 2.3) der Firma Union Rich Development[4].

Abbildung 2.3: Altina GBT-709

Dieses Gerät kann entweder per USB oder per Bluetooth mit einem Smartphone oder einem PDA verbunden werden.
In Rahmen dieser Arbeit wird der GPS Empfänger via Bluetooth mit dem Smartphone verbunden. Es stellt sich somit die Frage nach der Qualität respektive der Stabilität der Bluetooth-Verbindung und des GPS Empfangs. Tabelle 2.2 zeigt einige in späteren Experimenten sicherlich auftretende Situationen und ihre Auswirkungen auf die Verbindung und den Empfang von GPS Daten. Daraus wird ersichtlich, dass der für die Qualität der GPS-Daten entscheidende Faktor nicht die Bluetooth Verbindung mit dem Mobilephone, sondern eine möglichst gute Verbindung zu den GPS-Satelliten ist.

Einige weitere für diese Arbeit und spätere Experimente wichtige Daten sind nachfolgend noch kurz aufgeführt:

- Maximale Betriebszeit
  Mit dem integrierten, wiederaufladbaren Akku wird eine maximale Betriebszeit von ca. 5 Stunden erreicht.

- Akkuladezeit

[4] `http://www.altina.com.tw/`, Abrufdatum: 24.04.2007

Situation	Bluetooth	GPS Empfang
Mobilephone und GPS in Hosentasche	Ja	Gut
In Fahrzeug: Mobilephone in Hosentasche, GPS auf Beifahrersitz	Ja	Gut
Mobilephone und GPS in Aktentasche	Ja	Gut
Indoor: Mobilephone auf Tisch, GPS auf Fensterbank (Distanz ca. 2m)	Ja	Schlecht
Indoor: Mobilephone und GPS auf Tisch	Ja	Nein

Tabelle 2.2: GPS Verbindung

Das Aufladen des integrierten Akkus benötigt ca. 2 Stunden.

- Durchschnittliche Zeit für GPS Verbindungsaufbau
  Cold Start: weniger als 50 Sekunden
  Warm Start: weniger als 40 Sekunden
  Hot Start: weniger als 1.5 Sekunden

GPS Datenformat

Die GPS Daten werden an das verbundene Gerät im NMEA Format gesendet. Tabelle 2.3 zeigt den Aufbau eines GPS Datensatzes im NMEA Format.

`$GPGGA,002958.000,4741.4560,N,00839.3388,E,1,05,1.7,424.0,M,48.0,M,,0000*54`
`$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,b,q,nn,d.d,a.a,M,g.g,M,h.h,rrrr*hh`

Tabelle 2.3: Format der GPS Daten

Ein solcher Datensatz enthält die folgenden Informationen (Quelle: Wikipedia.org[5]):

- hhmmss.ss = Aktuelle Uhrzeit in Stunden (hh), Minuten (mm), Sekunden (ss) und Millisekunden (ss)
- llll.ll = Breitengrad
- a = Hemisphäre des Breitengrads: Norden (N) oder Süden (S)
- yyyyy.yy = Längengrad
- b = Hemisphäre des Längengrads: Osten (E) oder Westen (W)

[5] `http://de.wikipedia.org/wiki/NMEA`, Abrufdatum: 24.05.2007

- q = GPS Qualität: 0 = ungültig, 1 = GPS fix, 2 = DGPS fix, 6 = geschätzt (6 nur bei NMEA-0183 ab Version 2.3)
- nn = Anzahl der benutzten Satelliten (i. d. R. 0 bis 12)
- d.d = Horizontale Dilution of Precision (Verschlechterung der Position)
- a.a = Höhe der Antenne über Geoid
- M = Einheit der Antennenhöhe (Meter)
- g.g = Geoidal separation
- M = Units of geodidal separation (meters)
- h.h = Alter der DGPS-Daten
- rrrr = DGPS-Referenzstation (0000-1023)
- Prüfsumme

Detailliertere Informationen zu GPS und dem NMEA Format sind unter der angegebenen Internetadresse erhältlich.

2.2 Software

In diesem Kapitel soll auf die Software eingegangen werden, welche verwendet wurde um dies Arbeit durchzuführen. Es wird dabei zuerst die verwendete Entwicklungsumgebung vorgestellt. Danach soll kurz auf zusätzliche verwendete Software und Frameworks von Drittherstellern eingegangen werden.

2.2.1 Entwicklungsumgebung

Da die im Rahmen dieser Diplomarbeit entwickelte Software in C# geschrieben wurde und für den Einsatz auf Windows Mobile 5.0 Pocket PC vorgesehen ist, bot sich die Verwendung von Microsofts Visual Studio 2005 an.

2.2.2 Software / Frameworks von Drittherstellern

An dieser Stelle soll ein kurzer Überblick über die in dieser Arbeit verwendete Software, Frameworks und Libraries von Drittherstellern gegeben werden. Zum Einsatz kamen die in Tabelle 2.4 aufgeführten Komponenten:

Name	**Zweck**	**Hersteller**
OpenNETCF	Diverse Libraries	OpenNETCF.org [a]
WIMO Camera API	Library zum ansteuern der integrierten Kamera	Brian Cross [b]
GPS.NET API	Library für objektorientierten Zugriff auf GPS	GeoFrameworks [c]
MortSaver	Verhindert das aktivieren des Standby-Betriebs	Mirko Schenk [d]
MatLab	Transormation der Daten für die Experimentauswertung	The MathWorks [e]
Weka	Auswertung der Experiment-Daten	The University of Waikato [f]

Tabelle 2.4: Verwendete Software von Drittherstellern

[a] `http://www.opennetcf.org/home.ocf`, Abrufdatum: 25.04.2007
[b] `http://www.wimobot.com`, Abrufdatum: 25.04.2007
[c] `http://www.geoframeworks.com`, Abrufdatum: 25.04.2007
[d] `http://www.sto-helit.de`, Abrufdatum: 25.04.2007
[e] `http://www.mathworks.com/`, Abrufdatum: 25.04.2007
[f] `http://www.cs.waikato.ac.nz/ml/weka/`, Abrufdatum: 25.04.2007

3

Implementation

In diesem Kapitel soll auf die Implementation der Software eingegangen werden. Hierbei werden verschiedene Aspekte mit Einfluss auf die Architektur oder das Design der Software betrachtet. In einem ersten Schritt wird überprüft, inwiefern sich die verschiedenen internen und externen Datenquellen in die Software einbinden und verwenden lassen. Dabei soll die grundsätzliche Machbarkeit als auch allfällige Einflüsse auf die Performance des Gesamtsystems untersucht werden.

Im zweiten Teil dieses Kapitels wird der Aufbau und die Funktionsweise der Software vorgestellt. Der Fokus liegt hierbei auf der Funktionalität und dem Verhalten der Software.

3.1 Machbarkeitsanalyse

Im folgenden Abschnitt werden nun die verschiedenen Komponenten und Datenquellen hinsichtlich ihrer Verwendung und Anbindung an die Software geprüft. Diesem Arbeitsschritt kommt grosse Bedeutung zu, da es sich an dieser Stelle zeigen wird, welche Daten in den späteren Experimenten verwendet werden können. Es soll dabei auch analysiert werden, wie sich einzelne Komponenten auf die Performance der ganzen Software auswirken.

3.1.1 Anbindung der Datenquellen

Als erster Schritt der Machbarkeitsanaylse soll die Anbindung der einzelnen Datenquellen geprüft und aufgezeigt werden. Für die unter Kapitel 2.1 eingeführten Komponenten sowie interne Datenquellen des Palm Treo sollen dabei - sofern relevant - die folgenden Punkte untersucht werden:

- Physische Anbindung
- Software-seitige Anbindung
- Auswirkungen auf Stabilität der Software

Sensorboard

Die physische Anbindung des im Kapitel 2.1.2 beschriebenen Sensorboard erfolgt über einen TTL/UART kompatiblen Ausgang. Da dieses Board speziell für den Anschluss am Multi-Connector Port des Palm Treo entwickelt wurde, sind somit keine weiteren Modifikationen für die physische Verbindung nötig. Aus Gründen der Stabilität belegt das Board jedoch nicht nur den eigentlichen seriellen Eingang des Palm Treo, sondern ebenfalls den Eingang für das Aufladen des Akkus. Dies bedeutet, dass das Gerät während den Akkuladephasen nicht für das Sammeln von Daten verwendet werden kann.
Die Software-technische Anbindung des Sensorboards erfolgt über die von Microsoft im der .NET Compact Framework bereitgestellte Serial-API (genaue Bezeichnung einfügen). Die Klasse (Bezeichnung) stellt dabei die benötigte Lese- und Schreibfunktionalität für mit dem Mobilephone verbundene Serielle Geräte bereit.
Da das Sensorboard über keinen eigenen Timer verfügt, stellte sich die Frage, ob das auslesen der Daten in einem Push oder Pull Verfahren geschehen soll.

- Push-Verfahren
  In diesem Verfahren sendet das Sensorboard sämtliche Werte ungetaktet direkt nach dem Erfassen weiter an das Mobile Phone. Dies bedeutet, dass die Software den seriellen Port ständig überwachen und gegebenenfalls die hereinkommenden Daten vor der weiteren Verarbeitung zwischenspeichern muss. Die im Eingangspuffer liegenden Werte werden in konfigurierbaren Zeitabständen mit einem Zeitstempel versehen und gespeichert. Der Vorteil dieser Variante ist, dass sämtliche vom Sensorboard erfassten Werte auch durch die Software erfasst und gespeichert werden. Falls jedoch der Eingangspuffer der Software aufgrund einer hohen Systemauslastung zu viele Werte zwischenspeichern muss, werden die Werte beim Speichern mit einem minimal verschobenen Zeitstempel versehen.
- Pull-Verfahren
  Bei dieser Variante sendet die Software in konfigurierbaren Zeitabständen ein Signal zum Sensorboard, worauf dieses die aktuellen Messwerte zurückliefert. Der Vorteil dieser Variante liegt darin, dass die Werte in konfigurierbaren und konstanten Zeitabständen gemessen

werden. Da jedoch die minimale Zeiteinheit im .NET Compact Framework 500ms beträgt könnten bei dieser Variante lediglich zwei Messungen pro Sekunde durchgeführt werden. Da dies eindeutig zu wenig Messwerte ergeben würde, um beispielsweise ein Gehen des Benutzers zuverlässig zu erkennen, kam diese Variante nicht zum Einsatz.

Mit dem gewählten Push-Verfahren werden somit alle Werte, welche durch die Sensoren erfasst werden auch auf dem Gerät gespeichert. Das Sensorboard liefert derzeit ca. 15-20 Messwerte pro Sekunde. Durch diese hohe Dichte an Messwerten wird es später möglich sein, selbst komplexere Verhalten (wie beispielsweise Gehen) des Benutzers zu erkennen.

GPS

Der verwendete GPS Empfänger verfügt wie im Kapitel 2.1.3 beschrieben über eine Bluetooth und eine USB Schnittstelle. Da der Palm Treo über keinen USB Eingang verfügt kommt für die physische Anbindung des GPS folglich nur Bluetooth in Frage. Da jedoch Windows Mobile seit der Version 5.0 die Möglichkeit bietet, einen allfällig mit dem Gerät verbundenen GPS Empfänger unabhängig von der physischen Verbindung automatisch zu verwalten, spielt die physische Anbindung des GPS eine untergeordnete Rolle. Für die weitere Verwendung innerhalb einer Software kann nun direkt auf einen vom Betriebssystem bereitgestellten virtuellen COM-Port zugegriffen werden.
Für die software-seitige Anbindung wurde auf die GPS.NET API der Firma GeoFrameworks [1] zurückgegriffen. Dieses Package stellt verschiedene objektorientierte Klassen, Methoden und Events bereit, um auf die Daten eines mit dem Mobilephone verbundenen GPS Empfängers zuzugreifen.
Das Auslesen und Speichern der GPS Daten wird von dem selben Thread gesteuert, welcher das Serialboard verwaltet. Die GPS Daten werden in dem selben Buffer als auch in der selben Datei wie die Sensordaten des Serialboards abgelegt. Diese Methode stellt die Synchronizität der GPS Daten mit den durch die Sensoren erfassten Daten auf allen Schichten der Software sicher.

Audio

Da das im Mobilephone integrierte Mikrophon als Aufnahmegerät für die Audio Daten dient, entfällt an dieser Stelle das Problem der physischen Anbindung.
Zur Verwendung des Mikrophons und zur Aufzeichnung von Audiodaten kommt die OpenNet-

[1] `http://www.geoframeworks.com`, Abrufdatum: 25.04.2007

CF Media Library [2] zum Einsatz. Diese Library stellt verschiedene Möglichkeiten zum Aufzeichnen und Abspielen von Audio Dateien im wave Format zur Verfügung.
Die grösste Schwierigkeit im Zusammenhang mit dem Aufzeichnen von Audiodaten ergibt sich jedoch aus der Anforderung einen möglichst lückenlosen Stream über eine verhältnismässig lange Zeitperiode zu erfassen und zu speichern. Es gilt also, aus den beiden nachfolgend genannten Methoden die für den gegebenen Anwendungsbereich geeignetere zu finden.

- Aufzeichnung eines einzigen, zusammenhängenden Streams
  Bei dieser Methode werden die Audiodaten von der Software permanent aufgezeichnet und im Falle eines Events (z.B. Telefonanruf) die Aufnahme gestoppt. Nach Beendigung des Telefonanrufs wird die Aufnahme erneut gestartet. Dieses Verfahren hat jedoch einige auf den ersten Blick nicht auffallende negative Effekte. So lässt es sich bei dieser Methode nicht garantieren, wie lange die einzelnen Audio-Streams sein werden. Dies erschwert die Verknüpfung mit den Daten aus den anderen Sensoren, welche immer über eine identische Zeitspanne erfasst werden. Das grösste Problem bei dieser Methode sind jedoch die auf einem Mobilephone zur Verfügung stehenden Ressourcen, insbesondere die Grösse des RAM-Speichers. Durch das Aufzeichnen von sehr langen Audio-Streams kann es zu unkontrollierbaren Laufzeitfehlern oder gar Abstürzen des Mobilephones kommen.

- Aufzeichnung von mehreren kleineren Streams
  Bei diesem Verfahren wird im Gegensatz zu der oben beschriebenen Methode nicht ein einzelner Audio-Stream bis zum Auftreten eines Events aufgezeichnet, sondern stattdessen eine definierbare Anzahl von kleineren Streams, welche zusammengefasst die selbe Zeitspanne abdecken wie die von den restlichen Sensoren erfassten Daten. Dadurch wird die Verknüpfung der Daten aus den verschiedenen Quellen deutlich vereinfacht. Zudem lässt sich somit das Risiko von Laufzeitfehlern aufgrund von beispielsweise zu wenig Hauptspeicher auf ein Minimum reduzieren. Ein Nachteil dieses Verfahrens ist es, dass zwischen den einzelnen Streams eine Lücke entsteht, welche je nach Systemauslastung mehrere Sekunden dauern kann.

Für die Audio Anbindung wurde das zweite Verfahren gewählt, da allenfalls entstehende Lücken für den vorgesehenen Einsatzbereich weniger signifikant sind als die Nachteile der ersten Methode. Die Grafik 3.1 veranschaulicht das Speichern der einzelnen Streams in einem Ringbuffer auf Dateibasis.

[2] `http://www.opennetcf.org`, Abrufdatum: 25.04.2007

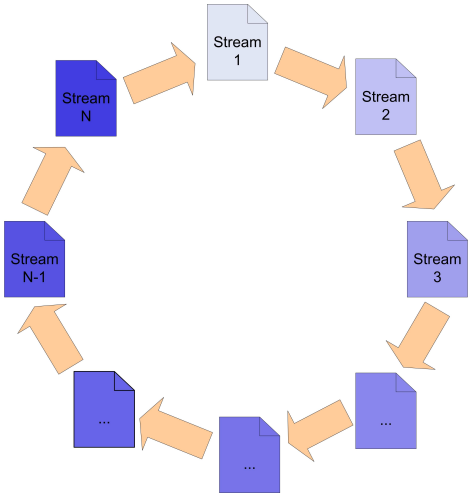


Abbildung 3.1: Audio Ringbuffer auf Dateibasis

Kamera

Das Verwenden der im Mobilephone integrierten Kamera stellt eine weitere Herausforderung dar. Aus der Common Language Runtime (CLR) des Compact Frameworks kann nicht direkt auf die integrierte Kamera zugegriffen werden, ohne dass für die Aufnahme eines Bildes oder eines Videos eine Bestätigung des Benutzers erfolgen muss. Für die Anbindung der Kamera wird darum wiederum eine Library eines Drittanbieters verwendet. Zum Einsatz kommt dabei die WiMo Camera API von Brian Cross [3]. Diese API ermöglicht es, unter C# die integrierte Kamera programmatisch auszulösen und das erfasste Bild weiter zu verarbeiten. Die Bilder werden nun innerhalb der Software von einem dedizierten Thread in konfigurierbarer Qualität und definierbaren Zeitabständen aufgenommen und gespeichert. Für die Speicherung der Bilder kommt wiederum ein Ringbuffer auf Datei Basis zum Einsatz.

Da jedoch beim Verwenden der Kamera regelmässig schwere Ausnahmen auftreten, welche die gesamte Software beenden, wurde von der weiteren Verwendung der integrierten Kamera für diese Arbeit abgesehen.

[3] `http://www.wimobot.com`, Abrufdatum: 25.04.2007

3.1.2 Telefonanrufe

Eine zentrale Anforderung an die Software ist es, auf eingehende Telefonanrufe zu reagieren. Da mit dieser Software auch andere, Event-basierte Experimente ohne weitere Anpassungen durchgeführt werden sollen, muss es auch möglich sein, Telefonanrufe in konfigurierbaren Abständen zu simulieren. Implementiert wurde dieser Simulationsmodus, indem die Schnittstelle, welche die möglichen Ereignisse im Zusammenhang mit einem Telefonanruf definiert abstrahiert (Klasse: `CallEventBroker`) und jeweils für echte Telefonanrufe (Klasse: `TelState`) und simulierte Events (Klasse: `Simulator`) implementiert wurde. Die wichtigsten Anforderungen in diesem Bereich sind die folgenden:

- Eingehende Anrufe erkennen
  Damit in einem späteren Experiment die gesammelten Sensordaten einem Anruf zugeordnet werden können und somit Untersuchungen über den Einfluss des Kontexts des Benutzers auf seine Unterbrechbarkeit gemacht werden können, ist es zwingend erforderlich, dass die Software eingehende Anrufe erkennt und verarbeiten kann. Es sollen sämtliche Daten, welche in einer definierbaren Zeitspanne vor diesem Zeitpunkt gesammelt wurden zusammengefasst und auf die Speicherkarte geschrieben werden.

- Ersetzen der optischen Anruf-Signalisation
  Damit die Probanden in späteren Experimenten Angaben zu ihrer Unterbrechbarkeit oder ihrem Kontext machen können, sobald ein Telefonanruf eingeht muss es möglich sein, die Standard Anrufsignalisation durch anpassbare Formulare zu ersetzen.

- Handling des eingehenden Telefonanrufs
  Nach Möglichkeit soll der Anruf basierend auf den Angaben des Benutzers in der Software direkt und programmatisch angenommen oder abgelehnt werden können.

Für das Erkennen von eingehenden Anrufen stellt das Compact Framework in der Version 2.0 die State and Notification Broker API (SNAPI) zur Verfügung, über welche man sich für auftretende Systemereignisse wie beispielsweise einen Telefonanruf registrieren kann. Tritt das überwachte Ereignis ein, informiert das Betriebssystem die Software, damit diese entsprechend reagieren kann. Mit diesem sehr einfach und komfortabel zu implementierenden Ansatz ist es jedoch nicht möglich, den Anruf programmatisch anzunehmen oder abzulehnen. Es ist auch nicht möglich, die Standardsignalisation des Anrufs zu entfernen oder auszublenden.

Die daraus entstehenden Probleme und deren Lösungen sind werden in den nächsten beiden Abschnitten detailliert beschrieben.

Handling der Telefonanrufe mit der Telephony API (TAPI)

Als Alternative zur SNAPI bot sich folglich der Weg über die Telephony API (TAPI) an, welche erweiterte Möglichkeiten bietet um Telefonanrufe zu steuern. Da jedoch diese API von Microsoft nur für die Verwendung aus einem in native Code (C++) geschriebenen Programm angeboten wird, musste eine wrapper Library für C# verwendet werden. Zum Einsatz kam dabei die TAPI Library von OpenNetCF [4]. Diese Library stellt die für diese Arbeit benötigten Objekte und Methoden zur Verwendung innerhalb einer in C# geschriebener Software zur Verfügung. Im Gegensatz zu der sehr komfortablen Observer-Architektur welche in der SNAPI zum Einsatz kommt ist jedoch einiges an Zusatzarbeit vonnöten um die oben genannten Anforderungen zu erfüllen.
Die verwendete TAPI Library sendet Status Meldungen an die Software, welche anschliessend analysiert und weiterverarbeitet werden müssen. Mit dieser Methode lassen sich jedoch bedeutend mehr Informationen zu einem eingehenden Telefonanruf extrahieren. Für den interessierten Leser sind im Anhang A die für diese Arbeit zentralen Folgen von TAPI Messages aufgeführt.

Das Parsen dieser Messages ermöglicht es folglich, auf eingehende Anrufe zu reagieren und sämtliche für diese Arbeit und spätere Experimente nötigen Informationen zu einem eingehenden Telefonanruf zu erhalten.

Ersetzen der optischen Anrufsignalisation

Durch den Einsatz der TAPI wird es nun möglich, Telefonanrufe aus der Software anzunehmen oder abzulehnen. Da jedoch die Standard Anrufsignalisation immer noch angezeigt wird, kann der Benutzer die von der Software angezeigten Formulare umgehen und den Anruf über den herkömmlich Weg entgegennehmen. Da die Anzeige des Anrufs zudem einen Drittel des Bildschirms bedeckt (siehe Bild 3.2), musste ein Weg gefunden werden, diese zu deaktivieren oder auszublenden. Die pragmatische Lösung dieses Problems besteht mangels Alternativen darin, die auf Windows Mobile Geräten als Standard definierte Anwendung zur Behandlung von Telefonanrufen (*cprog.exe*) bei einem eingehenden Telefonanruf zu beenden. Damit das Gerät danach wieder normal verwendet werden kann, wird diese Anwendung nach einem verpassten oder beendeten Anruf erneut gestartet. Dieses Verfahren führt jedoch zu zwei weiteren Problemen:

[4] `http://www.opennetcf.org`, Abrufdatum: 25.04.2007

- Telefongespräche können nicht mehr normal beendet werden.
  Da die Telefonapplikation von Windows Mobile nicht in Betrieb ist, kann ein Anruf nicht mehr wie gewohnt durch das Betätigen der Äufhängen-Taste beendet werden.

- Keine Ton- und Vibrations-Signalisation der Anrufe
  Durch das Beenden der Telefonapplikation entfällt sowohl die akkustische Signalisation als auch die Benachrichtigung durch den Vibrationsalarm.

Das Beenden eines aktiven Telefongesprächs wird darum von der Software wiederum durch den Einsatz der TAPI Library gesteuert. Die Anrufsignalisation durch einen Rufton und/oder den Vibrationsalarm wurde deshalb ebenfalls komplett neu implementiert.

Abbildung 3.2: Standard Anruf Signalisierung

Abbildung 3.3: Modifizierte Anruf Signalisierung

3.2 Software Design

In diesem Kapitel soll das Design der Software aufgezeigt werden. Zuerst wird dabei kurz auf die Architektur der Software eingegangen und die verschiedenen Software Komponenten beschrieben. Danach soll aufgezeigt werden, wie die Software bestimmte Ereignisse verarbeitet.

3.2.1 Statischer Aufbau(Aufbau/Architektur)

Da das .NET Framework das Observer Pattern durch das Sprachelement `Event` nativ unterstützt, lag es nahe, dass die Kommunikation zwischen den verschiedenen Softwarekomponenten mehrheitlich durch dieses Pattern implementiert wurde. Die Applikation besteht somit aus mehrheitlich lose gekoppelten Klassen, welche untereinander mittels Austausch von Events kommunizieren. Die zentralsten Klassen und ihre Zuständigkeiten sind in Tabelle 3.1 aufgeführt. Der Quellcode der gesamten Applikation kann auf der beiliegenden CD-ROM (siehe Anhang E) eingesehen werden.

Klasse	Beschreibung
`MainForm`	Diese Klasse startet und steuert die gesamte Applikation
`SerialDataLogger`	Die Klasse `SerialDataLogger` speichert die Sensordaten aus Sensorboard und GPS
`AudioRecorder`	Der `AudioRecorder` dient zur Aufnahme und Speicherung der Audio Streams
`PictureRecorder`	Die Klasse `PictureRecorder` steuert die periodische Aufnahme von Bildern durch die integrierte Kamera
`CallEventDataCollector`	Im `CallEventDataCollector` werden verschiedene, mit dem Anruf verbundene Daten gesammelt und gespeichert.
`CallEventBroker`	Die abstrakte Klasse `CallEventBroker` definiert die Schnittstelle für das Verwenden der Anruf-Spezifischen Events und Methoden.
`TelState`	Die Klasse `TelState` bildet die Implementation der im `CallEventBroker` definierten Schnittstelle für echte Anrufe.
`Simulator`	`Simulator` ist eine Implementation von `CallEventBroker` zur Generierung von simulierten Anrufen.
`WMCADTSystem`	Diese Klasse verwaltet die wichtigsten Einstellungen für das Gesamtsystem in Form von statischen Feldern, Events und Methoden.

Tabelle 3.1: Klassenübersicht

3.2.2 Dynamischer Aufbau (Ablauf)

Dieser Abschnitt behandelt den Ablauf der Software. Es soll dabei insbesondere erläutert werden, wie die Software auf Events (eingehende Anrufe) reagiert. Grafik 3.4 soll dabei eine Übersicht über die verschiedenen in diesem Prozess involvierten Komponenten geben und als Grundlage für die Diskussion dienen.

Der Thread, welcher die gesamte Applikation steuert, befindet sich in der Klasse `MainForm`. Dieser Thread ist zuständig für das Starten und Stoppen der weiteren Komponenten der Software. Nach dem Erkennen eines Events wird dieser entsprechend den Einstellungen durch Vibrationsalarm und gegebenfalls durch Rufton signalisiert und sämtliche Datensammlungs-Threads gestoppt. Danach werden dem Benutzer verschiedene Formulare für die Annotation des Events

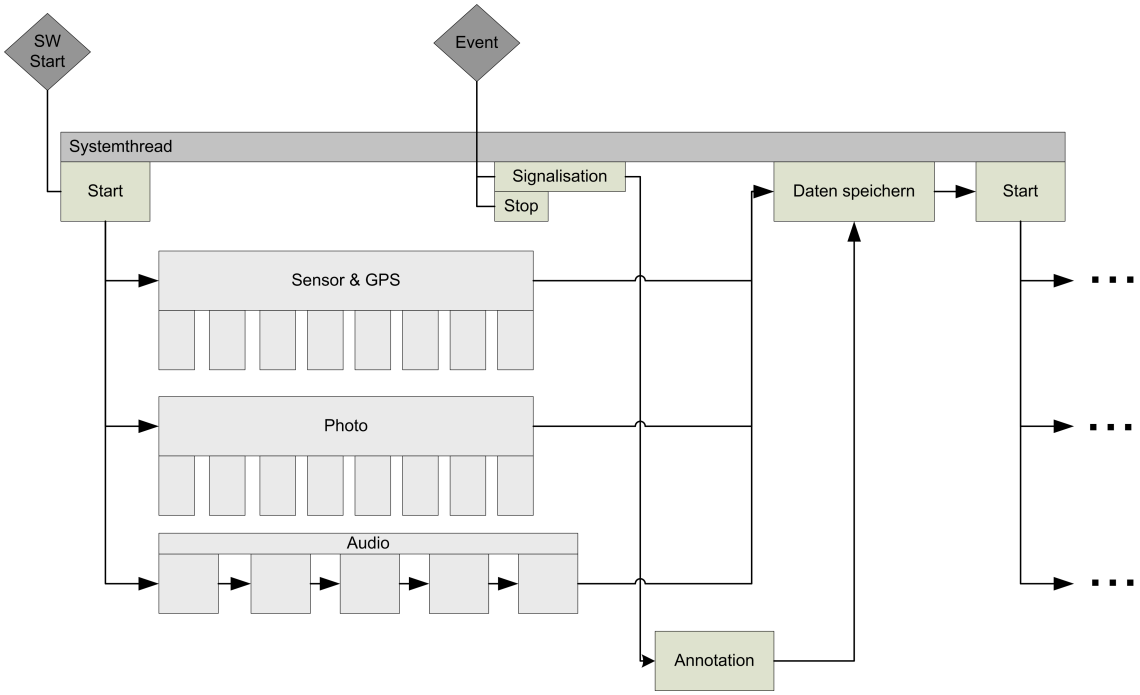


Abbildung 3.4: Software Ablauf

angezeigt. Abschliessend werden die gesammelten Daten auf die im Gerät eingesetzte SD-Speicherkarte transferiert. Nach der erfolgreichen Transferierung werden sämtliche Threads für die Datensammlung neu gestartet. Speziell zu erwähnen ist an dieser Stelle der Stoppvorgang für den Audio-Thread. Da es beim Abbrechen eines Audio-Aufnahmevorgangs zu einem Fehler kommt, welcher die gesamte Applikation beendet, musste hier eine alternative Lösung gefunden werden. Es wurde eine Lösung gewählt, welche den Audio-Thread nie stoppt, sondern stattdessen den aktuellen Stream fertig aufzeichnet und danach die Aufnahme in eine temporäre Datei umleitet. Bei dem Neustart wird die Ausgabe bei dem nächsten Stream im Ringbuffer fortgesetzt.

3.2.3 Datenformate, Speicherorte & Konfigurationsdatei

Da die Applikation diverse, unterschiedlich geartete Daten aufzeichnen muss, kommen für die Speicherung der Daten verschiedene Mechanismen und Formate zum Einsatz. Die Daten, welche durch das Serialboard und den GPS-Empfänger gesammelt werden, werden in einem Ringbuffer mit konfigurierbarer Länge im Hauptspeicher des Gerätes abgelegt. Nach dem Auftreten eines Events werden die Daten in eine CSV Datei in einem konfigurierbaren Daten-Verzeichnis (Standard: */My Documents/WMCADT*) im Unterordner */serial* abgelegt. Die Audio- und Bilddaten werden direkt nach dem Erfassen ebenfalls im vogegebenen Daten-Verzeichnis in den Unterordnern */audio* respektive */pictures* gespeichert. Für die Audio Daten wird das wav-Format verwendet. Die Bilder werden als JPG gespeichert. Die zusätzlich zu einem Event erfassten Daten wie beispielsweise Annotationen durch den Benutzer oder aktuelles Datum und Uhrzeit werden in einer Textdatei im Daten-Verzeichnis gespeichert.
Sämtliche Einstellungen für die Applikation können in der Datei *settings.txt* im Hauptverzeichnis der Applikation geändert werden. Die Datei *settings.txt* wird im Anhang B genauer beschrieben.

4

Evaluation

In diesem Kapitel wird das im Rahmen dieser Arbeit durchgeführte Experiment beschrieben und die Resultate besprochen. Dabei wird zuerst der Aufbau und der Ablauf des Experiments beschrieben. Danach sollen kurz die Methoden und Hilfsmittel erwähnt werden, welche verwendet wurden, um die aus dem Experiment erhaltenen Daten zu untersuchen. Der Hauptteil dieses Kapitels ist jedoch den eigentlichen Ergebnissen und den daraus gewonnen Erkenntnissen gewidmet.

Durch dieses Experiment soll nun ermittelt werden, ob es möglich ist, anhand von Daten aus verschiedenen mit dem Mobiltelefon verbundenen Sensoren Aussagen über die Aktivität und die Umgebung des Benutzers machen zu können. Dabei sollen vor allem die folgenden Punkte untersucht werden:

- Vorhersage-Qualität der einzelnen Sensoren
- Geeignete Algorithmen
- Qualität von Sensorkombinationen
- Optimaler zeitlicher Vorlauf der Datenerfassung

4.1 Experiment-Aufbau

Für die Durchführung dieses Experiment kam die im Kapitel 3 beschriebene Applikation zum Einsatz. Anstatt jedoch die Daten jeweils bei einem eingehenden Telefonanruf zu speichern, wurden einige Änderungen an der Software vorgenommen, welche es ermöglichen, solche Telefonanrufe zu simulieren. Für dieses Experiment wurde jeweils in Intervallen von fünf Minuten ein solcher Anruf simuliert und der Benutzer aufgefordert, Angaben zu seiner aktuellen Aktivität und seinem Aufenthaltsort zu machen. Das mit dieser Konfiguration der Software ausgerüstete Mobiltelefon wurde von der Testperson während einer Periode von 21 Tagen während alltäglichen Aktivitäten mitgeführt. Auf diese Weise entstand ein Datensatz von 261 Events. Dieser Datensatz dient als Grundlage für die nachfolgenden Auswertungen und Diskussionen. Die Testperson im nachfolgenden Experiment war der Autor dieser Arbeit.

Die Nachfolgenden Abschnitte beschreiben die einzelnen Schritte, welche den in dieser Arbeit verwendeten und in Grafik 4.1 visualisierten KDD Prozess (Knowledge Discovery in Databases) bilden. Im Abschnitt 4.2 werden die verwendeteten Rohdaten beschrieben. Abschnitt 4.3 befasst sich mit dem Preprocessing der Daten. Das Data Mining sowie die Analyse der Resultate folgt schliesslich im Abschnitt 4.4.

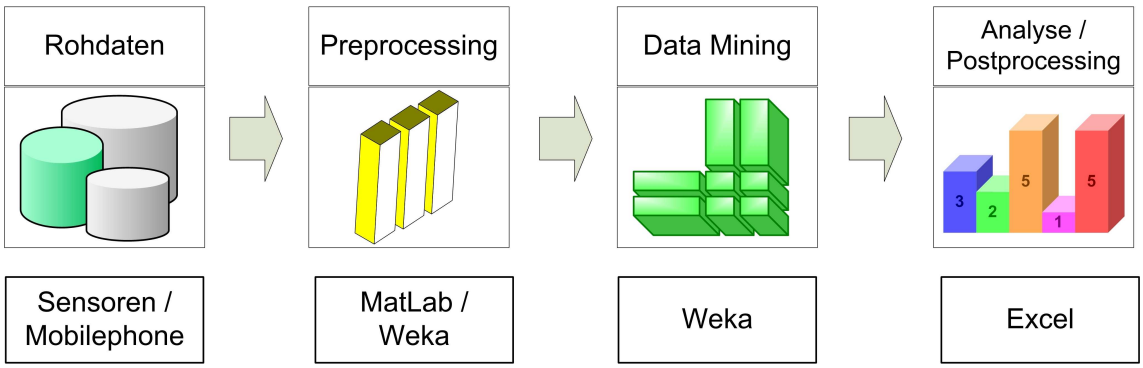


Abbildung 4.1: KDD Prozess. Die Abbildung zeigt die einzelnen Schritte des durchgeführten KDD Prozesses sowie die jeweilige Quelle der Daten respektive die zur Aufbereitung verwendeten Tools.

4.2 Deskriptive Statistik

An dieser Stelle sollen die für die nachfolgenden Auswertungen verwendeten Daten beschrieben werden. Für die Auswertungen stehen sämtliche Daten zur Verfügung, welche Durch die Sensoren am Mobilephone erfasst wurden. Da es mit der Implementierung der Kamera technische Schwierigkeiten gab, stehen jedoch keine Bilddaten zur Verfügung.

Die Sensordaten, welche für die Nachfolgenden Untersuchungen zur Verfügung stehen sind in Tabelle 4.1 aufgeführt. Da die Sensordaten über einen Zeitraum von fünf Minuten vor einem Event gesammelt wurden, müssen die im jeweils gewählten Intervall liegenden Werte pro Sensor zu einem einzelnen Feature zusammengefasst werden. In dieser Arbeit geschieht dies jeweils dadurch, dass für die Werte im jeweiligen Intervall der Durchschnitt (mean) und die Standardabweichung (std) berechnet wird.

Sensor	**Quelle**	**Beschreibung**
Temperatur	Serialboard	Umgebungstemperatur des Mobilephones
Accelerometer	Serialboard	Beschleunigung des Mobilephones (triaxial)
Gyroskop	Serialboard	Winkelbeschleunigung de Mobilephones (triaxial)
Gas	Serialboard	Gemessene Gase: CO & Alkohol
Magnet	Serialboard	Magnetfeldsensoren (triaxial)
Audio	integriertes Mikrophon	Umgebungsgeräusche des Mobilephones
Audio Features	Audio	Dedizierte Aufschlüsselung der Umgebungsgeräusche in 17 verschiedene Audio Features. Eine detaillierte Beschreibung dieser Features und ihrer Bedeutung findet sich in [Egger, 2004]
GPS	Externes GPS	Geographische Position des Mobilephones
Signalstärke	Mobilephone	Stärke des GSM Empfangs
Hour of day	Mobilephone	Tageszeit

Tabelle 4.1: Verfügbare Sensordaten

Der zur Verfügung stehende Datensatz umfasst 261 Instanzen. Grafik 4.2 zeigt die einzelnen Klassen für das Attribut *Activity* und ihre Häufgkeit innerhalb der Verteilung. Dieselben Informationen finden sich für das Attribut *Location* in der Grafik 4.3. Die Rohdaten wurden für die nachfolgenden Auswertungen nicht verändert. Die einzigen Änderungen betreffen die Korrektur

von offensichtlichen Tippfehlern und wo aus dem Zusammenhang ersichtlich, das Ergänzen von fehlenden Eingaben.

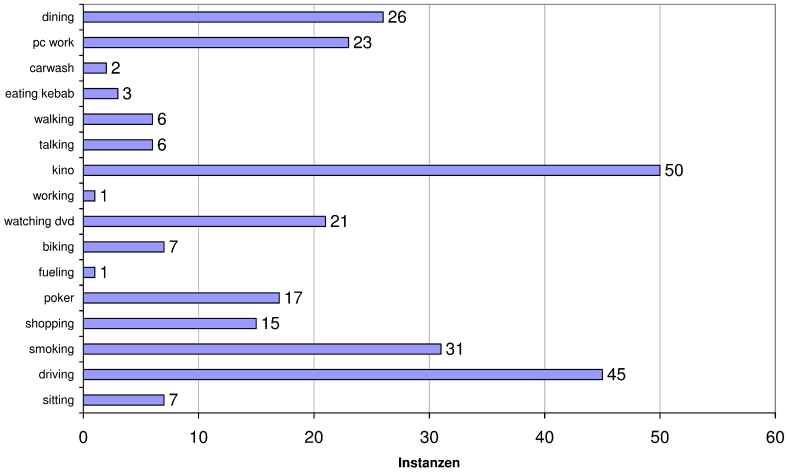


Abbildung 4.2: Ausprägungen Activity

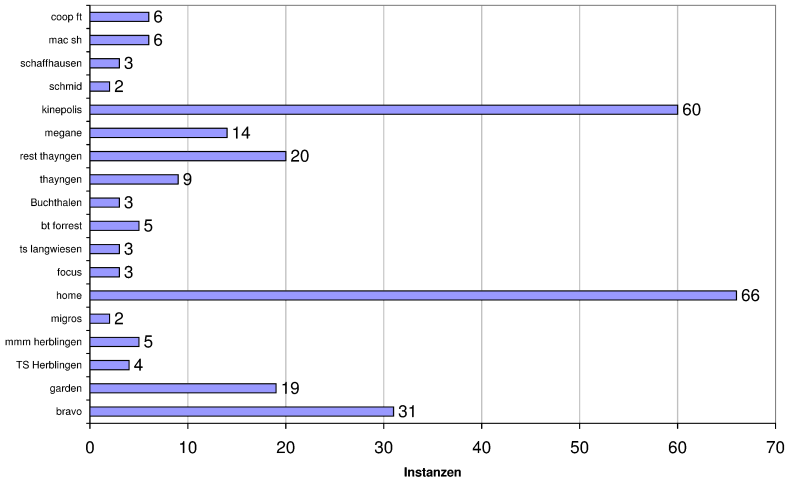


Abbildung 4.3: Ausprägungen Location

4.2.1 Prior

Der Prior stellt für die nachfolgende Besprechung einen wichtigen Wert dar. Er zeigt den durch eine zufällige Voraussage maximal erreichbaren Wert und bildet somit die Messlatte für andere Voraussagemethoden. Es soll darum kurz aufgezeigt werden, wie er berechnet wird und welchen Wert er für die vorliegenden Daten annimmt.

Der Prior lässt sich berechnen, in dem man aus einer Verteilung die am häufigsten auftretende Klasse Klasse nimmt. Der Prozentuale Anteil dieser Klasse an der Gesamtzahl der Verteilung wird Prior genannt. Für die verwendeten Daten ergeben sich für den Prior somit die folgenden Werte:

- Activity: 50/ 261 = 0.19157 = 19.16%
- Location: 66 / 261 = 0.25287 = 25.29%

Falls man also neben der Klassenverteilung keine weiteren Informationen zur Verfügung hätte und darum für jeden Wert in der Verteilung annehmen würde, dass es sich um die Klasse mit der häufigsten Ausprägung handelt, würde man bei der Aktivität in 19.16% der Fälle und bei der Location in 25.29% der Fälle richtig liegen.
Damit ein Algorithmus aussagekräftig ist, muss er mindestens dem Prior entsprechen. Liegen die Werte darunter haben die Ergebnisse keine Aussagekraft.

4.3 Vorbereitung der Daten

Die nachfolgenden Auswertungen der gesammelten Daten wurden mit der Data Mining Software **Weka 3** (University of Waikato Environment for Knowledge Analysis) gemacht. Weka ist eine Software, welche verschiedene Algorithmen für das bearbeiten von Data Mining Aufgaben beinhaltet. Weka ermöglicht auch das Preprocessing der Daten, wie zum Beispiel das Filtern der verschiedenen Attribute oder auch das Normalisieren der Werte gewisser Attribute. Nährere Informationen über Weka finden sich auf der Internetseite des Herstellers [1].
Die Daten wurden dabei zuerst in MatLab[2] präpariert und in das von Weka verwendete ARFF Format gebracht. Für den Zweck dieser Diplomarbeit wird für jeden Sensor jeweils über sämtliche vorhandenen Werte innerhalb der gewählten Zeitspanne der Durchschnitt sowie die Standardabweichung berechnet und weiterverwendet. Diese verhältnismässig einfache Zusammenfassung der einzelnen Werte hat den Vorteil, dass sie auch in späteren Experimenten für allfällige Realtime-Auswertungen direkt auf dem Mobilephone verwendet werden könnte. Die einzige Ausnahme bildet hier das Attribut *Audio*, für welches mehrere Features generiert wurden. Es wurde ebenfalls die Standardabweichung und der Durchschnitt berechnet. Diese Features werden für die weiteren Auswertungen als *Audio* bezeichnet. Zusätzlich wurden jedoch noch pro Sekunde 17 weitere Features aus den Audio-Daten generiert und wiederum unter dem Durchschnitt und der Standardabweichung zusammengefasst. Es sind dies die folgenden Features:

- Spectral center of gravity
- Temporal fluctuations of spectral center of gravity
- Common onsets across frequency bands
- Histogram width
- Variance
- Mean level fluctuations strength
- Zero crossing rate
- Cepstral coefficients 1-10

[1] `http://www.cs.waikato.ac.nz/ml/weka/` , Abrufdatum: 12.05.2007
[2] `http://www.mathworks.com` , Abrufdatum: 30.05.2007

Eine genauere Beschreibung der einzelnen Features findet sich in [Egger, 2004].

Die Skripte, welche für das Preprocessing der Daten zum Einsatz kamen, sowie eine Anleitung zur Verwendung derselben finden sich auf der beigelegten CD-ROM (siehe E) respektive im Anhang C.

Die auf diese Weise aus den Rohdaten generierten Features sowie Angaben zu den jeweiligen Wertebereichen für eine Intervalllänge von fünf Minuten sind in den Tabellen 4.2 (mean) & 4.3 (std) ersichtlich.

Attribut	**Min**	**Max**	**Mean**	**Std**
Temperatur	2190.601	4090	2735.085	587.659
Accelerometer x-Achse	1817.135	3787.732	2126.385	456.623
Accelerometer y-Achse	1880.947	3786.211	2126.191	456.826
Accelerometer z-Achse	1887.129	3790.851	2214.274	431.017
Gyroskop x-Achse	2031.105	2461.298	2131.453	139.058
Gyroskop y-Achse	1813.666	2149.756	2067.314	113.712
Gyroskop z-Achse	2034.128	2462.355	2132.4	135.94
Gas 1	656.34	2432.222	779.753	162.844
Gas 2	633.757	907.282	708.917	27.476
Magnet x-Achse	641.936	2490.279	1894.886	413.996
Magnet y-Achse	549.498	2171.851	1477.981	306.306
Magnet z-Achse	611.964	2253.105	1727.943	425.794
Audio	-0.01	-0.004	-0.004	0.001

Tabelle 4.2: Attribute (mean) und deren Wertebereiche

Attribut	Min	Max	Mean	Std
Temperatur	0	988.513	200.904	299.098
Accelerometer x-Achse	9.425	968.753	158.73	268.814
Accelerometer y-Achse	6.573	935.71	151.386	268.55
Accelerometer z-Achse	9.833	982.374	153.814	258.121
Gyroskop x-Achse	3.004	283.05	86.875	82.913
Gyroskop y-Achse	3.486	301.929	66.325	78.328
Gyroskop z-Achse	3.466	282.005	86.178	80.062
Gas 1	3.312	549.347	44.927	89.292
Gas 2	5.21	392.076	16.514	27.541
Magnet x-Achse	8.082	1708.807	396.366	497.888
Magnet y-Achse	8.007	1778.475	397.8	471.561
Magnet z-Achse	7.474	1658.423	384.507	479.454
Audio	0.005	0.612	0.088	0.113

Tabelle 4.3: Attribute (std) und deren Wertebereiche

4.4 Analyse der Resultate

In diesem Kapitel sollen die Resultate des Experiments präsentiert und analysiert werden. Im Abschnitt 4.4.1 werden dabei als erstes die einzelnen Sensoren, respektive Sensorgruppen hinsichtlich ihrer Voraussagequalität untersucht. Mit den daraus gewonnen Erkenntnissen soll danach versucht werden, optimale Kombinationen von Sensoren und optimale Zeitspannen zu finden und weiter zu analysieren. Die Ergebnisse aus diesem Arbeitsschritt finden sich im Kapitel 4.4.2. Für diese Auswertungen kommt dabei der Algorithmus J48 in Weka zum Einsatz. Der J48 Algorithmus erscheint als Decision Tree Algorithmus vor allem aus Gründen der Einfachheit des generierten Modells als Besonders geeignet. Ein solches Modell wäre auch für Folgeexperimente direkt auf dem Mobilephone implementierbar. Der J48 Algorithmus ist die Java Implementation des Decision Tree Algorithmus C4.5. Weitere Informationen zum C4.5 Algorithmus finden sich in [Quinlan, 1993].

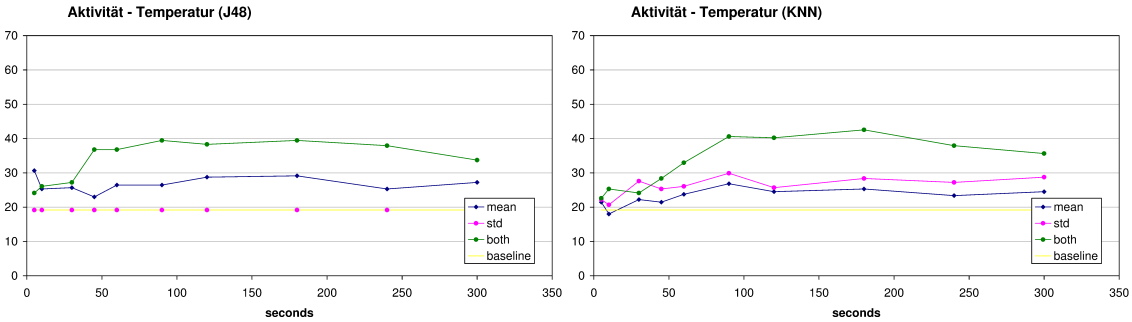
Als Vergleichsalgorithmus wurde ein K-nearest neighbour Algorithmus gewählt. Zum Einsatz kam dabei der Algorithmus IBk in Weka. Weitere Informationen zu diesem Algorithmus finden sich in [Aha and Kibler, 1991]. Für die nachfolgenden Auswertungen wurde mit den jeweils drei

nächsten Nachbarpunkten gerechnet.

4.4.1 Vorhersage-Qualität der einzelnen Sensoren

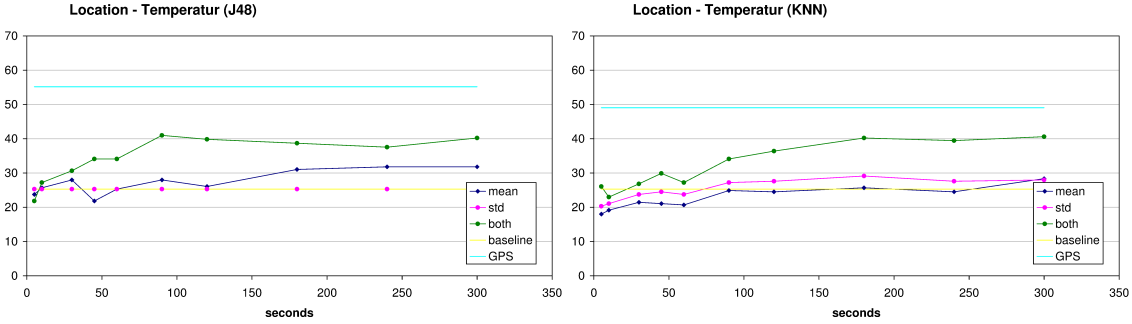
Die nachfolgenden Diagramme zeigen die Voraussagequalität der einzelnen Sensoren bezüglich Aktivität und Aufenthaltsort. Die x-Achse zeigt dabei die Zeit vor dem Event (in Sekunden). Auf der y-Achse kann der Voraussagewert in Prozent abgelesen werden. Als Vergleichswert ist in den Diagrammen jeweils der Prior eingefügt. Dieser Wert wird mit *baseline* bezeichnet. Für die Location-Diagramme ist als zusätzliche Referenz der vom GPS erreichte Wert eingefügt. Dieser Wert wurde erzielt, indem jeweils die letzten bekannten GPS-Werte vor einem Event ausgewertet wurden. Es wurden auch Auswertungen vorgenommen, welche den Durchschnitt und die Standardabweichung der GPS Daten in einer gegebenen Zeitspanne verwendeten. Weitere Auswertungen wurden mit dem absolut letzten Wert vor einem Event durchgeführt, welcher bei fehlendem Empfang innerhalb von Gebäuden 0 entspricht. Keine dieser Auswertungen übertraf jedoch den mit der letzten bekannten Position erreichten Wert.

Der GPS Wert erscheint auf den ersten Blick dennoch ziemlich niedrig. Dies lässt sich unter anderem damit erklären, dass der Aufenthaltsort nicht immer zwingend geografisch bezeichnet wurde (Beispiel: Auto). Vor allem aber führt der fehlende GPS Empfang innerhalb von Gebäuden dazu, dass diese Aufenthaltsorte zum grössten Teil nicht voneinander unterschieden werden können.



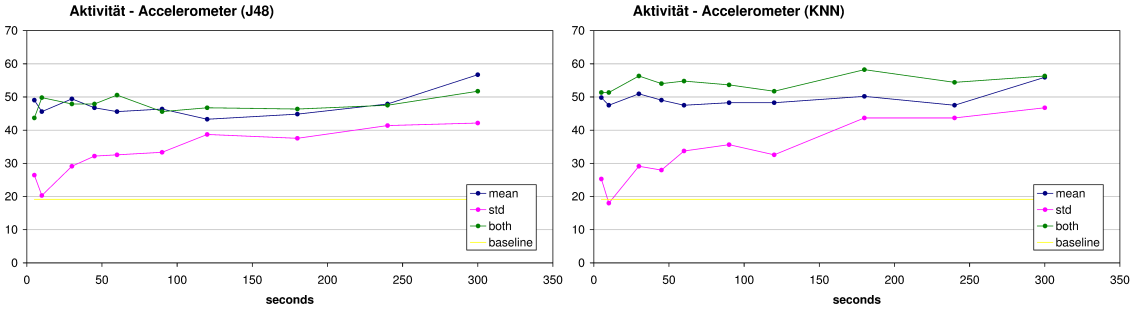
(a) J48 (b) KNN

Abbildung 4.4: Die Vorhersagewerte basierend auf der Umgebungstemperatur übertreffen den Prior (baseline) nur knapp (mean) respektive gar nicht (J48-std). Interessant erscheint jedoch, dass sich die erzielbaren Werte ab einer Intervallänge von 45 Sekunden stark verbessern, wenn sowohl der Durchschnitt als auch die Standardabweichung miteinbezogen werden. Im Weiteren fällt auf, dass die Werte ab einer Intervalllänge von mehr als 180 Sekunden wieder abnehmen.



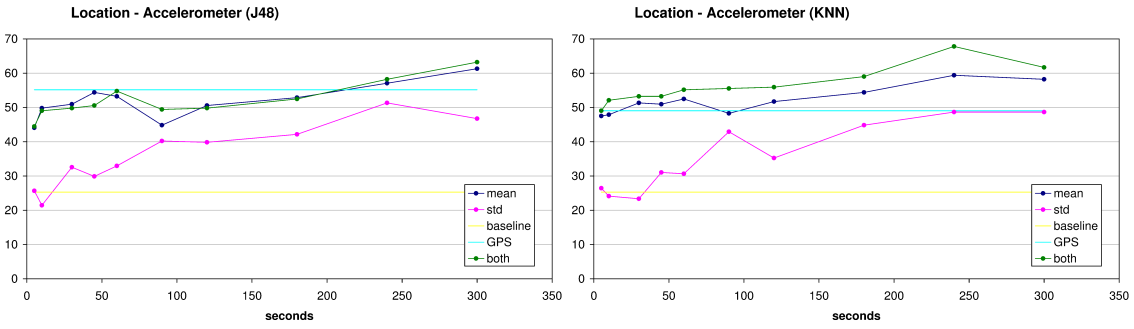
(a) J48 (b) KNN

Abbildung 4.5: Die Voraussagewerte für den Aufenthaltsort basierend auf der Umgebungstemperatur weisen ein ähnliches Muster auf wie jene für die Aktivität. Die Werte übertreffen den Prior nur knapp (mean) respektive gar nicht (std). Auch für den Aufenthaltsort verbessern sich die Werte stark, wenn sowohl der Durchschnitt als auch die Standardabweichung miteinbezogen werden.



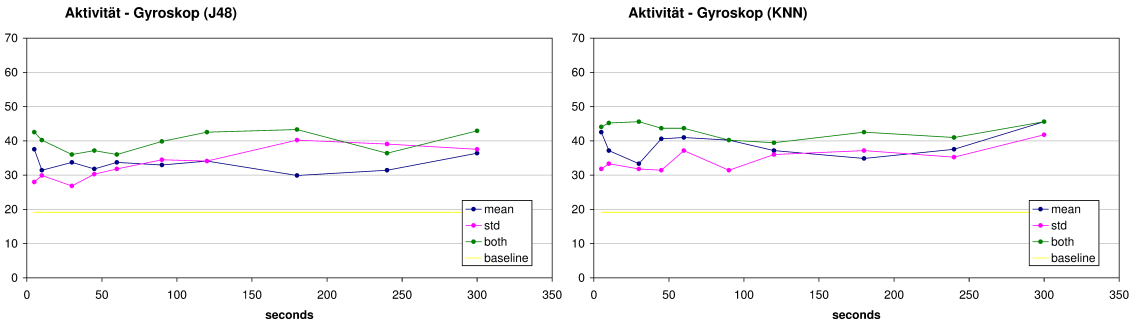
(a) J48 (b) KNN

Abbildung 4.6: Der Prior (baseline) wird sowohl durch den Durchschnitt (mean) als auch durch die Standardabweichung(std) übertroffen. Es lässt sich erkennen, dass die Voraussagequalität bei der Standardabweichung mit zunehmender Länge des beobachteten Intervalls ansteigt. Die Werte für den Durchschnitt sind dagegen über die gesamte Messreihe ziemlich stabil. Werden die beiden Features zusammen betrachtet, ergibt sich für den KNN Algorithmus eine lichte Verbesserung der Werte.



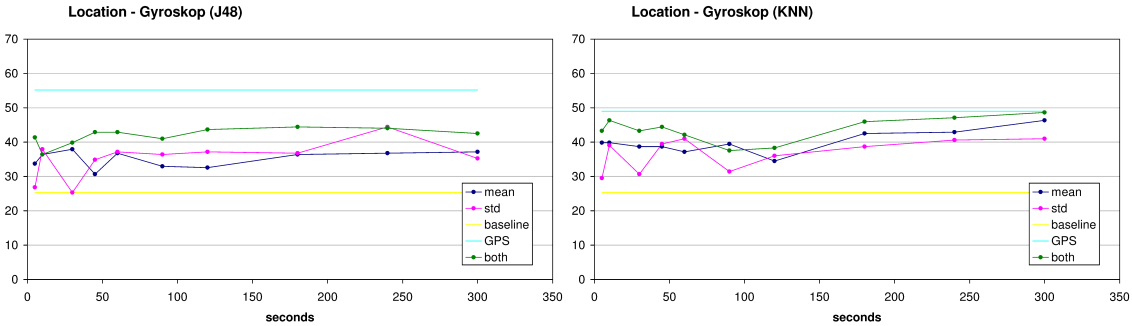
(a) J48 (b) KNN

Abbildung 4.7: Auch für den Aufenthaltsort lässt sich sowohl mit dem Durchschnitt als auch mit der Standardabweichung der Prior (baseline) übertreffen. Die Standardabweichung scheint jedoch sowohl für den J48, als auch für den KNN Algorithmus noch instabiler zu sein als bei der Voraussage der Aktivität. Mit dem KNN Algorithmus werden jedoch vor allem für die Kombination beider Features (both) deutlich bessere und stabilere Werte erreicht. So liegen diese Werte beim KNN praktisch von Beginn an über dem GPS-Benchmark, während sie beim J48 erst ab einer Intervalllänge von 240 Sekunden darüber liegen.



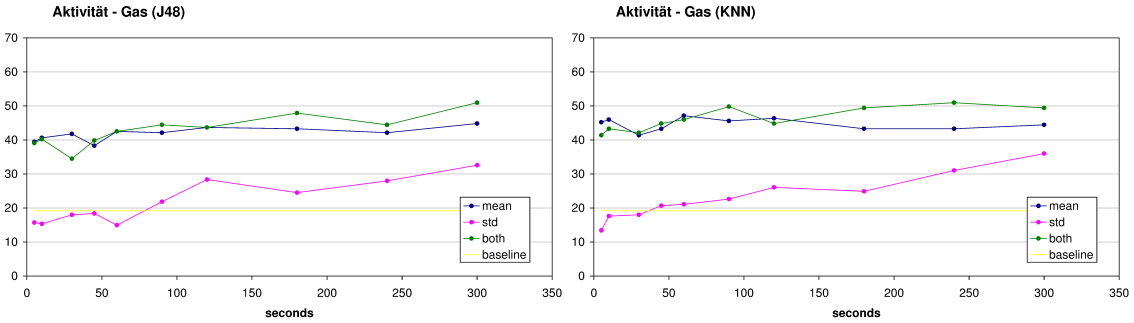
(a) J48 (b) KNN

Abbildung 4.8: Der Prior wird auch hier übertroffen. Die Voraussagequalität erreicht jedoch nicht die vom Accelerometer erzielten Werte. Die Werte bewegen sich für beide Algorithmen im selben Bereich und erscheinen ebenfalls für beide Algorithmen sehr instabil.



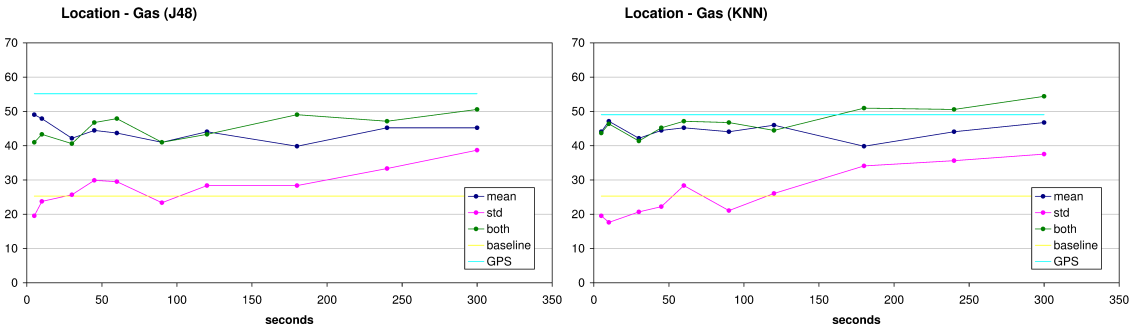
(a) J48 (b) KNN

Abbildung 4.9: Auch für den Aufenthaltsort wird der Prior übertroffen. Jedoch wird der GPS Benchmark von keinem Feature erreicht. Die Instabilität dieses Sensors wird für den Aufenthaltsort noch deutlicher sichtbar. Die Stabilität lässt sich jedoch durch eine Kombination beider Features (both) etwas verbessern.



(a) J48 (b) KNN

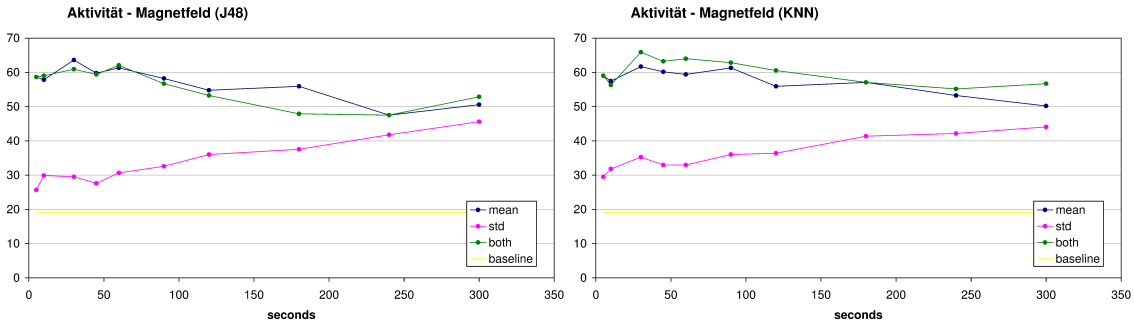
Abbildung 4.10: Sowohl für den Durchschnitt als auch für die Standardabweichung (ab 90 Sekunden für J48, ab 60 Sekunden für KNN) wird auch mit den Gas-Sensoren der Prior (baseline) übertroffen. Auch für diese Sensoren scheint der Durchschnitt zuverlässiger und besser geeignet als die Standardabweichung. Die Ergebnisse erscheinen für den Durchschnitt und die Kombination (both) sehr stabil. Allerdings lässt sich das Ergebnis durch die Kombination von Durchschnitt und Standardabweichung im Vergleich zu den bisherigen Sensoren nicht wesentlich verbessern.



(a) J48 (b) KNN

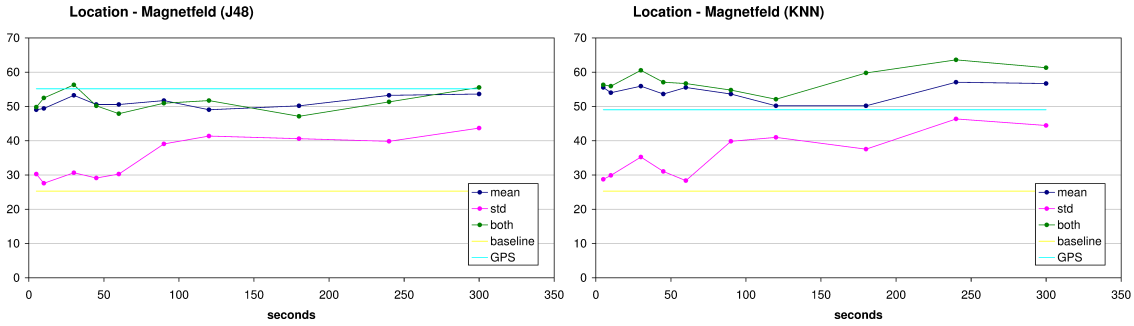
Abbildung 4.11: Die Voraussagewerte für den Aufenthaltsort unterscheiden sich nur gering von jenen für die Aktivität. Der GPS Benchmark wird knapp nicht erreicht (J48), respektive ab 180 Sekunden leicht überschritten (KNN). Die beiden Algorithmen liefern ansonsten keine nennenswert unterschiedlichen Resultate.

Anmerkung: Die beiden verwendeten Gas-Sensoren wurden an dieser Stelle zusammen ausgewertet, da sie sich gut ergänzen. Im Anhang D finden sich noch getrennte Auswertungen für den CO-Sensor und den Alkohol-Sensor.



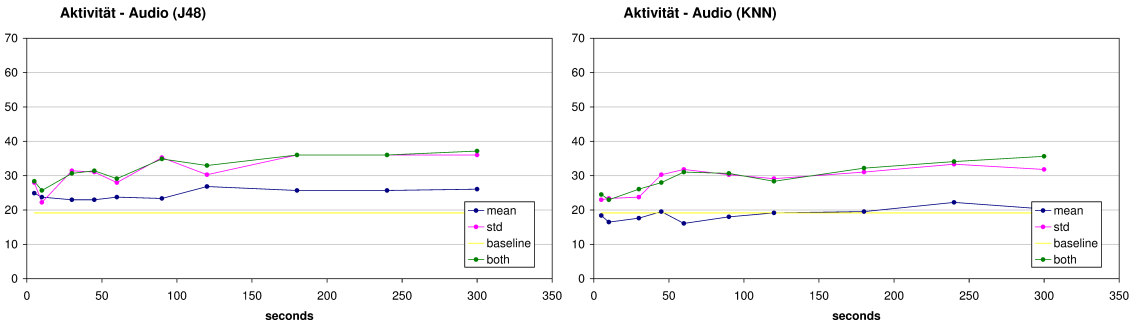
(a) J48 (b) KNN

Abbildung 4.12: Der Prior wird vor allem durch den Durchschnitt deutlich übertroffen. Auch die durch die Standardabweichung erzielten Werte liegen für alle Messungen über dem Prior. Interessant erscheint für diesen Sensor vor allem, dass die Qualität der Voraussage basierend auf dem Durchschnitt und der Kombination mit zunehmender Länge des Intervalls abnimmt, während sie für die Standardabweichung - wie von den anderen Sensoren bekannt - zunimmt. Die Kombination der beiden Features ergibt für diesen Sensor nur beim KNN Algorithmus eine leichte Verbesserung der Voraussagewerte.



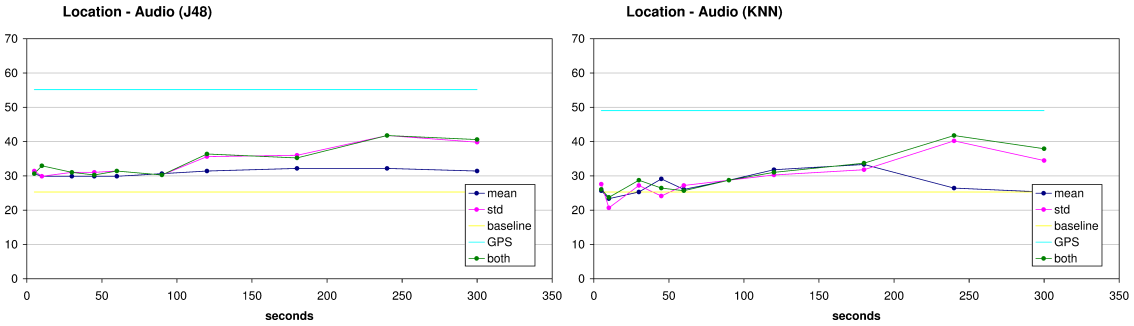
(a) J48 (b) KNN

Abbildung 4.13: Auch für den Aufenthaltsort wird der Prior mit diesem Sensor, vor allem mit dem Durchschnitts-Feature deutlich übertroffen. Die Standardabweichung liefert, verglichen mit den Werten für die Aktivität etwas weniger stabile Werte. Der GPS Benchmark wird bei den Werten, welche der KNN-Algorithmus liefert durch den Durchschnitt und die Kombination beider Features leicht übertroffen.



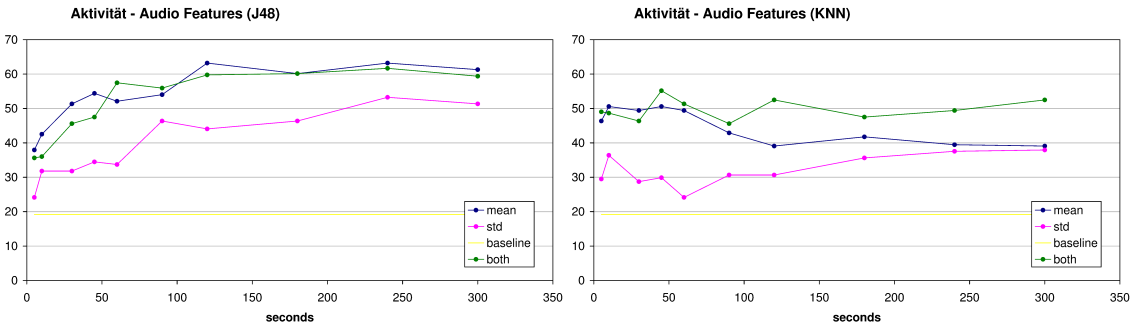
(a) J48 (b) KNN

Abbildung 4.14: Der Prior wird hier leicht übertroffen. Für die Audiodaten erscheint die Standardabweichung das deutlich bessere Feature zu sein. Die Qualität der Voraussage ist jedoch verglichen mit den bisher betrachteten Sensoren weit unterdurchschnittlich.



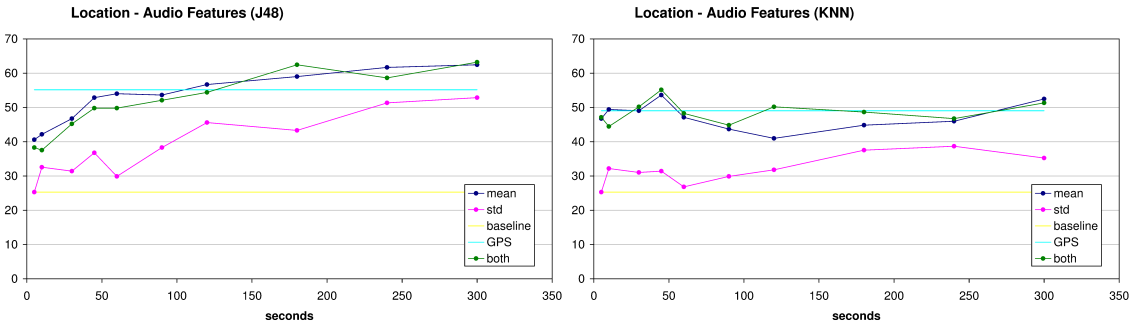
(a) J48 (b) KNN

Abbildung 4.15: Für den Aufenthaltsort zeigt sich in etwa das selbe Bild wie für die Aktivität. Jedoch sind die Werte für alle Features bis zu einer Länge des Intervalls von 90 Sekunden (J48) respektive 180 Sekunden (KNN) in einer sehr engen Spanne.



(a) J48 (b) KNN

Abbildung 4.16: Durch die dedizierte Untersuchung der Audiodaten lässt sich die Qualität der Voraussage erheblich steigern. Interessant erscheint hier, dass die Werte für den J48 Algorithmus bis zu einer Intervalllänge von 120 Sekunden deutlich ansteigen. Danach scheinen sie ziemlich stabil zu bleiben.



(a) J48 (b) KNN

Abbildung 4.17: Auch für den Aufenthaltsort lässt sich das Ergebnis durch die Verwendung der 17 verschiedenen Audio-Features deutlich verbessern. Der GPS Benchmark wird ab einer Intervalllänge von 120 Sekunden (J48) übertroffen. Der KNN Algorithmus erreicht wie schon bei der Aktivität nicht die vom J48 Algorithmus erzielten Werte. Dies dürfte darin begründet sein, dass die Wertebereiche der einzelnen Features trotz Normalisierung zu unterschiedlich sind, um mit den jeweils drei nächsten Nachbarn in einen 17-dimensionalen Raum gute Ergebnisse zu erzielen.

Diskussion

Zusammenfassend und für die weiteren Auswertungen lässt sich ausgehend von den bisherigen Ergebnissen sagen, dass die besten Ergebnisse bei einer Intervallänge von über 180 Sekunden erzielt werden. Eine Ausnahme bilden die Magnetfeldsensoren. Bei diesen Sensoren werden die besten Ergebnisse bei einer Intervallänge von weniger als 60 Sekunden erzielt. Der Magnetfeldsensor liefert auch die mit Abstand besten Werte.
Bezüglich der untersuchten Algorithmen lässt sich feststellen, dass sich die erzielten Werte in einer ähnlichen Bandbreite befinden. Der Ibk Algorithmus scheint allerdings etwas stabilere Ergebnisse zu liefern.

Die Tabellen 4.4, 4.5, 4.6 & 4.7 zeigen nochmals eine Übersicht mit dem jeweils besten erzielten Wert pro Sensor respektive pro Sensorgruppe und Algorithmus. Die Verallgemeinerung dieser Werte wurde jedoch noch nicht nachgewiesen, da es sich lediglich um die Resultate einer Messreihe handelt. Sie sollen daher eher eine grobe Übersicht geben, welche Werte etwa mit den gegebenen Parametern zu erwarten sind.

Sensor	**Intervall**	**Feature**	**Wert**
Temperatur	90&180 Sekunden	both	39.46%
Accelerometer	300 Sekunden	mean	56.70%
Gyroskop	180 Sekunden	both	43.29%
Gas	300 Sekunden	both	50.96%
Magnetfeld	30 Sekunden	mean	63.60%
Audio	300 Sekunden	both	37.16%
Audio Features	120/240 Sekunden	both/mean	63.22%

Tabelle 4.4: Beste Voraussagewerte bezogen auf Aktivität (J48)

Sensor	Intervall	Feature	Wert
Temperatur	90 Sekunden	both	41%
Accelerometer	300 Sekunden	both	63.22%
Gyroskop	180/240 Sekunden	both/std	44.44%
Gas	300 Sekunden	both	50.57%
Magnetfeld	30 Sekunden	both	56.32%
Audio	240 Sekunden	both/std	41.76%
Audio Features	300 Sekunden	both	63.22%

Tabelle 4.5: Beste Voraussagewerte bezogen auf Aufenthaltsort (J48)

Sensor	Intervall	Feature	Wert
Temperatur	180 Sekunden	both	42.53%
Accelerometer	180 Sekunden	both	58.24%
Gyroskop	30/300 Sekunden	both/mean	45.59%
Gas	240 Sekunden	both	50.96%
Magnetfeld	30 Sekunden	both	65.90%
Audio	300 Sekunden	both	35.63%
Audio Features	45 Sekunden	both	55.17%

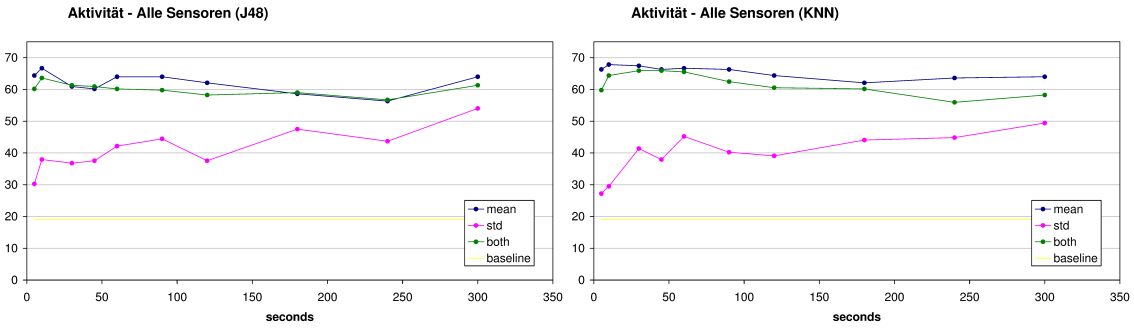
Tabelle 4.6: Beste Voraussagewerte bezogen auf Aktivität (KNN)

Sensor	Intervall	Feature	Wert
Temperatur	180 Sekunden	both	40.23%
Accelerometer	300 Sekunden	both	67.81%
Gyroskop	300 Sekunden	both	48.66%
Gas	300 Sekunden	both	54.41%
Magnetfeld	240 Sekunden	both	63.60%
Audio	240 Sekunden	both	41.76%
Audio Features	45 Sekunden	both	55.17%

Tabelle 4.7: Beste Voraussagewerte bezogen auf Aufenthaltsort (KNN)

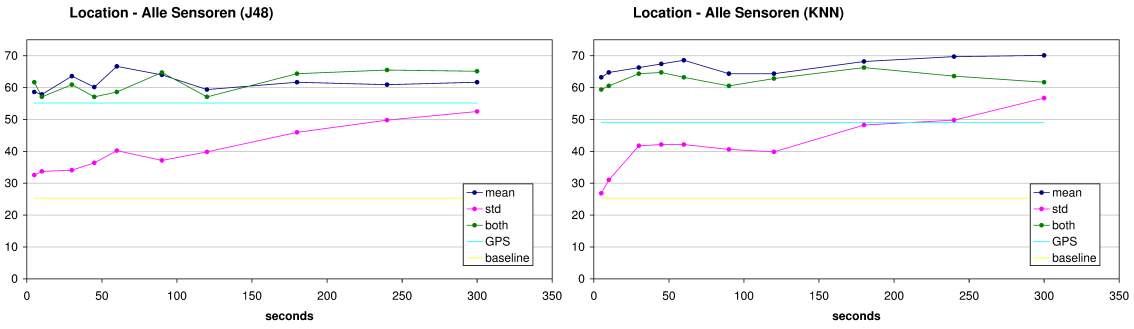
4.4.2 Vorhersage-Qualität von Sensorkombinationen

In diesem Abschnitt sollen nun verschiedene Sensorkombinationen untersucht werden. Das Ziel besteht darin, ein optimales Ergebnis mit möglichst wenigen Sensoren zu erreichen. In einem ersten Schritt wird untersucht, welche Werte erzielt werden können, wenn sämtliche Sensoren des Sensorboards benutzt werden. Das dabei erzielte Ergebnis dient als Grundlage für die weiteren Untersuchungen und Optimierungen welche danach durchgeführt werden. Das Ergebnis für die Kombination aus allen Sensoren des Sensorboards - wiederum basierend auf dem J48 und IBk Algorithmus - ist in den Grafiken 4.18 & 4.19 visualisiert.



(a) J48 (b) KNN

Abbildung 4.18: Aktivität - Alle Sensoren des Sensorboards



(a) J48 (b) KNN

Abbildung 4.19: Location - Alle Sensoren des Sensorboards

Durch das Verwenden einer Kombination aus allen Sensoren des Sensorboards lässt sich wie in Grafik 4.18 & 4.19 die Qualität der Vorhersage verglichen mit dem besten Wert eines einzelnen Sensors (Magnetfeld) von Werten im Bereich von 60% auf Werte im Bereich von 70% steigern.

Verglichen mit den Resultaten der einzelnen Sensoren bringt jedoch die Kombination (both) von Durchschnitt (mean) und Standardabweichung (std) keine wesentliche Verbesserung der Voraussagewerte. Insbesondere für den KNN Algorithmus wird deutlich, dass die besten Werte mit dem Durchschnittswert erzielt werden.

Nachfolgend soll nun untersucht werden, ob sich dieses Ergebnis durch eine Veränderung der Sensorkombination verbessern lässt und wie der Einfluss einzelner Sensoren auf das Gesamtergebnis ist. Es können dabei weitere Features hinzukommen, wie beispielsweise Audio oder auch allenfalls das Gesamtergebnis negativ beeinflussende Features weggelassen werden. Die gewählte Länge des Intervalls soll für diese Untersuchung so weit wie möglich aufgrund der bisherigen Erkenntnisse für jeden Sensor optimiert werden. Tabelle 4.8 zeigt die verwendeten Sensoren sowie die jeweils verwendete Intervalllänge.

Sensor	**Verwendete Länge des Intervalls**
Temperatur	180 Sekunden
Accelerometer	300 Sekunden
Gyroskop	240 Sekunden
Gas	300 Sekunden
Magnet	30 Sekunden
Audio	300 Sekunden
Audio Features	240 Sekunden
GPS	300 Sekunden & letzter Wert
Signalstärke	letzter Wert
Hour of day	letzter Wert

Tabelle 4.8: Verwendete Sensoren und Intervalllängen

Idealkombination & Relevanz der Features

In diesem Abschnitt wird untersucht, mit welcher Sensorkombination der beste Voraussagewert erzielbar ist und wie viel die einzelnen Sensoren zum Ergebnis beitragen. Die nachfolgenden Resultate beruhen wiederum auf den bereits bekannten Algorithmen J48 & Ibk (KNN mit drei Nachbarn).

Ausgehend von der Kombination aller Sensoren soll die Relevanz der darin enthaltenen Features ermittelt werden. Um die Relevanz einzelner Features zu determinieren, wurde ein *backward elimination* Verfahren angewendet ([Devijver and Kittler, 1982]). Aus der Kombination wird nun jeweils das am wenigsten zum Gesamtergebnis beitragende Feature entfernt. Weitere Informationen zur Selektion von relevanten Features finden sich in [Kohavi and John, 1997]. Die Ergebnisse dieses Arbeitsschrittes wurden in den Abbildungen 4.20 bis 4.23 visualisiert.

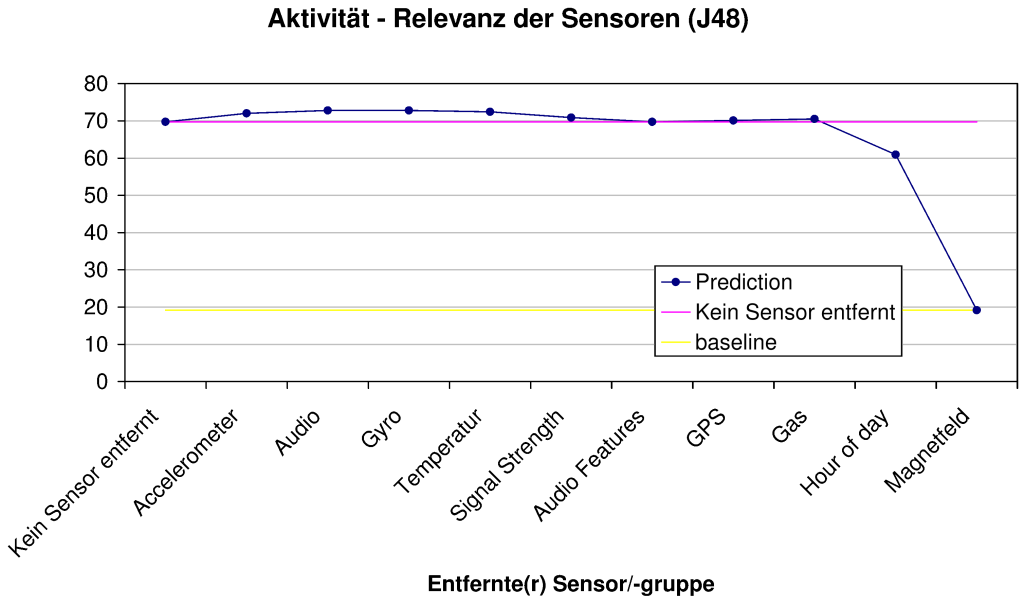


Abbildung 4.20: Wird kein Sensor aus der Kombination entfernt, beträgt der Wert 69.73 %. Das Entfernen von Accelerometer, Audio (Standardabweichung & Durchschnitt) sowie Gyroskop führt zu einer Verbesserung des Voraussagewerts auf 72.8%. Die verbleibenden Features bis Hour of day haben keinen starken Einfluss auf das Gesamtergebnis. Den grössten Beitrag mit einer Verbesserung gegenüber dem Prior leisten die Magnetfeldsensoren mit 41.76%.

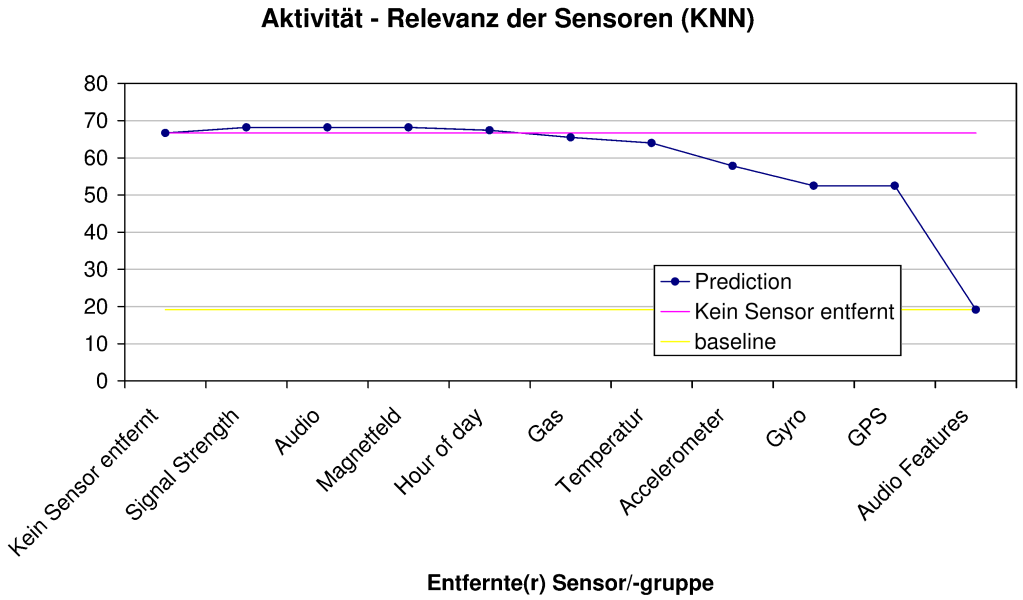


Abbildung 4.21: Werden alle Sensoren verwendet, beträgt der erzielte Wert 66.67%. Durch das Entfernen von Signal Strength, Audio (Standardabweichung & Durchschnitt) sowie den Magnetfeldsensoren verbessert sich das Gesamtergebnis leicht um 1.53% auf 68.2%. Im Vergleich zum J48 Algorithmus fällt beim KNN auf, dass bedeutend mehr Sensoren zum Gesamtergebnis beitragen. Im Weiteren liegen die erzielten Werte unter jenen, welche durch den J48 Algorithmus erzielt wurden. Interessant erscheint auch, dass sich die Reihenfolge der Relevanz der Features verglichen mit den Ergebnissen des J48 deutlich verändert hat.

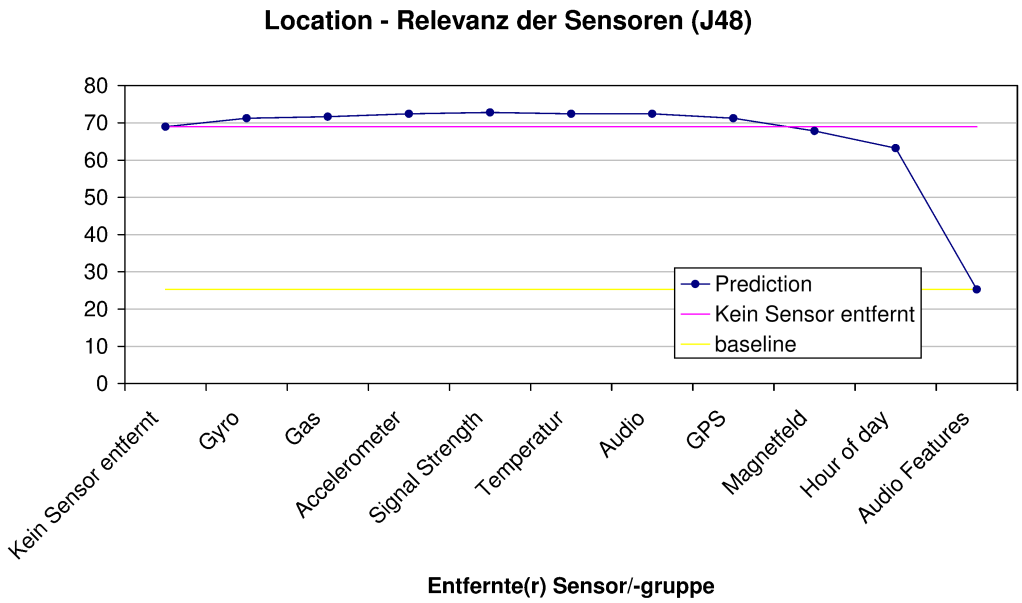


Abbildung 4.22: Der für alle Sensoren erzielte Wert beträgt 68.97%. Dieser Wert verbessert sich mit der Entfernung von Gyroskop, Gas-Sensoren, Accelerometer & Signal Strength auf 72.8%. Danach bleibt der Wert bis zum Entfernen der Magnetfeldsensoren ziemlich stabil (Bandbreite ca. 1%). Den grössten Beitrag leisten die Audio Features mit einer Verbesserung von 37.93% gegenüber dem Prior.

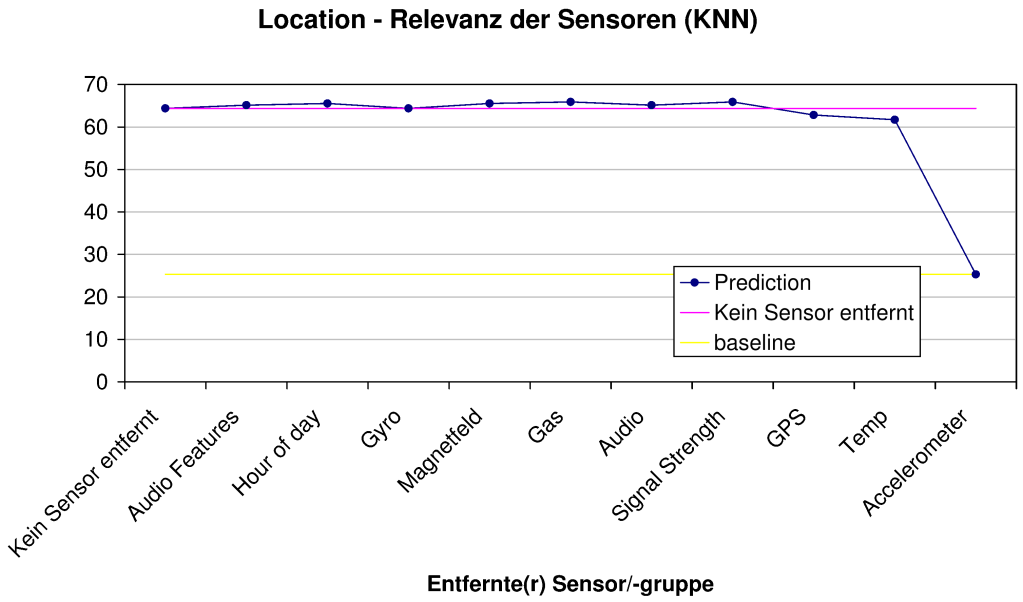


Abbildung 4.23: Der mit allen Sensoren erzielte Wert beträgt 64.37%. Das Entfernen der Features bis zu GPS lässt den Wert in einer sehr engen Spanne zwischen 65.90% und 64.37% schwanken. Erst das Entfernen des GPS führ zu einer deutlichen Verschlechterung des Ergebnisses. Den grössten Beitrag, eine Verbesserung gegenüber dem Prior von 36.4%, steuern hier die Beschleunigungssensoren bei. Die Werte erreichen jedoch auch für den Aufenthaltsort nicht die vom J48 Algorithmus erzielten.

Diskussion

Aufgrund dieser Analyse lassen sich leider noch keine allgemeingültigen Aussagen über die ideale Kombination von Sensoren machen. Es hat sich jedoch gezeigt, dass eine Kombination, welche aus allen verfügbaren Sensoren besteht, in keinem der Fälle das beste Ergebnis liefert. Weiter hat sich gezeigt, dass die meisten der Sensoren das Gesamtergebnis lediglich schwach beeinflussen und der grösste Beitrag durch eine Kombination von drei bis vier Sensoren geleistet wird.

4.4.3 Zusammenfassung der Ergebnisse

In den vorangegangenen Abschnitten wurden verschiedene Analysen gemacht. Es wurde zuerst untersucht, wie gut sich einzelne Sensoren eignen, um die Aktivität oder den Aufenthaltsort vorauszusagen. Danach wurden die Features aufgrund der Ergebnisse hinsichtlich des zeitlichen Vorlaufs vor dem Auftreten des Events optimiert. Aus den nun hinsichtlich der Intervalllänge optimierten Features wurde untersucht, welche Kombination die besten Ergebnisse liefert und wie die einzelnen Sensoren sich auf das Gesamtergebnis auswirken.

Zum Abschluss sollen nun die bisherigen Ergebnisse noch einmal zusammengefasst und verglichen werden. Die Abbildungen 4.24 & 4.25 zeigen dabei die folgenden Werte:

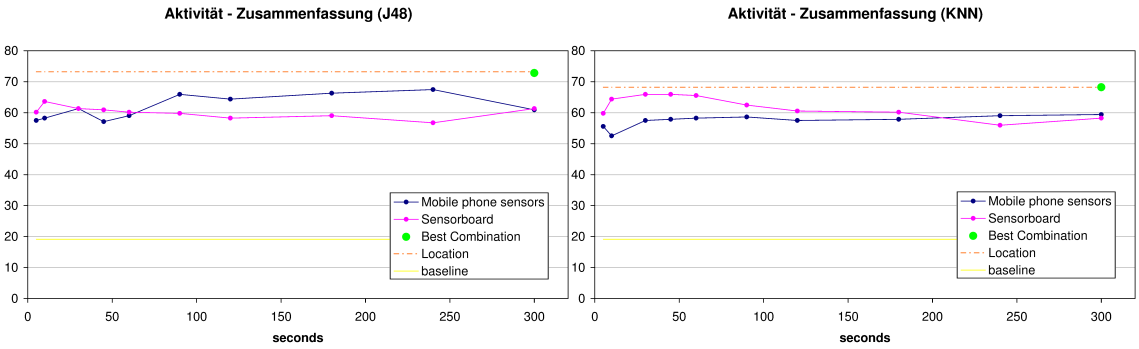
- **Mobile phone sensors**
  Eine Kombination von Sensoren, welche auf herkömmlichen Mobilephones ohne zusätzliche Sensoren (ausgenommen GPS) realisierbar wäre. Die Kombination umfasst: Audio, Time of Day, Signal Strength und GPS. Die dedizierten Audio Features kommen in dieser Kombination nicht zum Einsatz, da deren Berechnung auf einem Mobilephone so lange dauern würde, dass die Ergebnisse nicht instantan verwendet werden könnten.

- **Sensorboard**
  Eine Kombination aus allen Sensoren, welche sich auf dem Sensorboard befinden (siehe Abschnitt 4.4.2).

- **Best Combination**
  Der beste, in den vorangegangenen Auswertungen erzielte Wert (siehe Abschnitt 4.4.2).

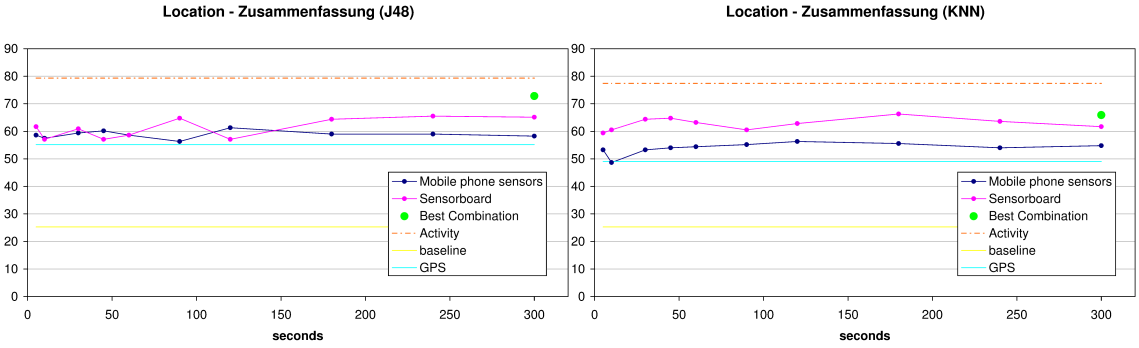
- **Location respektive Activity**
  Die Vorhersage der Aktivität durch den Aufenthaltsort (für Activity) respektive Vorhersage des Aufenthaltsortes durch die Aktivität (für Location).Diese Werte zeigen die Korrelation zwischen Aktivität und Aufenthaltsort. Da der verwendete Datensatz lediglich 261 Instanzen umfasst, zeigt sich hier eine hohe Korrelation. Diese Korrelation erklärt auch die ähnlichen Ergebnisse für die Aktivität und den Aufenthaltsort in den vorangegangenen Auswertungen.

- **GPS Benchmark & Prior (baseline)**



(a) J48 (b) KNN

Abbildung 4.24: Die Ergebnisse, welche durch die Sensoren des Sensorboards erzielt werden und die durch reine Mobilephone-Sensoren erzielbaren Werte liegen etwa im selben Bereich. Interessant erscheint, dass bei beiden Algorithmen anfänglich die Werte der Sensorboardkombination höher liegen und sich die Kurve danach mit jener der Mobilephone-Kombination kreuzt. Der Voraussagewert der besten Kombination deckt sich mit der Voraussage der Aktivität durch den Aufenthaltsort.



(a) J48 (b) KNN

Abbildung 4.25: Für den Aufenthaltsort werden durch die Kombination der Sensoren des Sensorboards werden etwas bessere Werte erzielt, als durch die Kombination der Mobilephone-Sensoren. Für den KNN Algorithmus scheinen die Sensoren des Mobilephones keinen grossen Einfluss zu haben, da der Wert der besten Kombination bereits durch die Sensorboard-Kombination erreicht wird. Beide Kombinationen schlagen jedoch bereits für sich genommen den GPS-Benchmark. Der J48 Algorithmus erzielt merklich bessere Ergebnisse.

Zusammenfassend lässt sich sagen, dass das in dieser Arbeit durchgeführte Experiment erste, vielversprechende Resultate geliefert hat. Einige Sensoren erreichen schon für sich alleine genommen sehr gute Voraussagewerte im Bereich von ca. 60% (z.B. Magnetfeld-Sensoren), sowohl bezüglich Aktivität, als auch für den Aufenthaltsort. Durch eine Kombination von geeigneten Sensoren kann die Qualität der Voraussage sogar auf über 70% gesteigert werden.

Es gibt jedoch einen Punkt, welcher beim Betrachten der Resultate miteinbezogen werden muss. Das in dieser Arbeit durchgeführte Experiment war vergleichsweise (zu einer real-world Anwendung) klein und wurde von nur einer Person durchgeführt. Aus diesem Grund ist die verfügbare Datenbasis ebenfalls vergleichsweise klein. Dieser Umstand äussert sich auch in der sehr hohen Korrelation von über 70% zwischen Aktivität und Aufenthaltsort. Dieser hohe Wert kommt vor allem dadurch zustande, dass während dem Experiment viele Tätigkeiten am immer gleichen Ort durchgeführt wurden. Sehr deutlich ist dieser Umstand bei der Aktivität ‘Kino’ zu erkennen, welche während dem Experiment immer am selben Ort (Kinepolis) stattfand.

Wie bereits erwähnt, sind die Resultate nichtsdestotrotz äusserst vielversprechend und dürften eine gute Grundlage bilden für weitere und umfangreichere Experimente.

5

Schlussfolgerung

In dieser Diplomarbeit wurde gezeigt, dass es möglich ist, mit den verwendeten Sensoren Aussagen über den Kontext eines Mobiltelefonbenutzers zu machen. Es wurde dazu eine Software implementiert und in einem kleineren Experiment einem ersten Praxistest unterzogen. Die Durchführung dieses Experiments hat gezeigt, dass die in dieser Diplomarbeit entwickelte Software geeignet ist, sämtliche benötigten Daten zu erfassen und zu speichern.

5.1 Limitations

Leider kommt auch die vorliegende Arbeit nicht ohne Einschränkungen aus. Die wichtigsten Einschränkungen sollen nachfolgend im Hinblick auf Folgearbeiten und/oder Experimente aufgeführt werden. Wo dies möglich ist, soll noch ein Ansatzpunkt für die Behebung oder Umgehung der Einschränkung aufgeführt werden.

5.1.1 Technische Einschränkungen

Die wichtigsten Technischen Einschränkungen sind die die nachfolgend aufgeführten Punkte.

- Akkulaufzeit
  Die maximale Laufzeit des Mobilephones mit aktivierten Gas-Sensoren beträgt lediglich ca. drei Stunden. Damit ein länger andauerndes Experiment sinnvoll durchgeführt werden kann, können die Gas-Sensoren bei Bedarf in den Einstellungen der Software deaktiviert werden (siehe Anhang B). Möglich ist auch eine Einstellung, bei welcher diese Sensoren

lediglich kurz vor einem Event aktiviert werden. Bei echten Telefonanrufen würde dies einer Aktivierung beim Eingang des Anrufs entsprechen. Da jedoch die Gas-Sensoren eine Aufwärmzeit von ca. 20 bis 30 Sekunden benötigen, ergäbe eine solche Einstellung in diesem Fall keine brauchbaren Werte. Eine Alternative wäre, die Gas-Sensoren periodisch zu aktivieren. Im Weiteren wäre es für länger andauernde Versuche sicher sinnvoll mit mindestens einem zusätzlichen Akku (sowohl für Mobilephone als auch für GPS) zu arbeiten.

- Speicher
  Je nach Einstellungen und der Anzahl von auftretenden Events kann der verfügbare Speicher auf der SD-Karte knapp werden. In dieser Arbeit wurde eine SD-Karte mit 512 MB Kapazität benutzt. Dies hatte zur Folge, dass die Karte nach ca. drei Stunden voll war und die Daten auf einen PC transferiert werden mussten. Für länger andauernde Experimente, insbesondere mit technisch weniger versierten Probanden ist es darum zwingend erforderlich mit grösseren Speicherkapazitäten zu arbeiten.

- Kamera
  Im Rahmen dieser Arbeit wurde auch die Verwendung der integrierten Kamera in der Software implementiert. Leider war es jedoch in der begrenzten Zeit nicht möglich, die in diesem Zusammenhang auftretenden Fehler zu beheben.

- Akkustische Anrufsignalisation
  Es wurde eine akkustische Anrufsignalisation implementiert, welche für das im Rahmen dieser Arbeit durchgeführte Experiment ausreichend war. Diese Implementation ist jedoch für längerfristige Experimente mit Probanden, welche das Mobilephone täglich nutzen nicht ausreichend, da der (als .wav-Datei hinterlegte) Rufton nicht wiederholt wird. Die Implementation der Anrufsignalisation kann bei Bedarf in der Klasse `Signaller` geändert werden.

- Standby
  Da sämtliche aktiven Programme pausiert werden, wenn das Mobilephone in den Standby-Modus wechselt, wurde im Rahmen dieser Arbeit ein eine zusätzliche Software verwendet, welche den Standby-Betrieb verhindert und gleichzeitig die Bildschirmbeleuchtung und die Tasten deaktiviert. Es bleibt jedoch darauf zu achten, dass das Mobilephone nie in den Standbybetrieb wechselt sei dies nun automatisch (lässt sich durch entsprechende Geräteeinstellungen verhindern) als auch manuell.

5.2 Ausblick

In dieser Arbeit wurde eine Software entwickelt, welche als Fundament und Ausgangspunkt für weitere Arbeiten und Experimente dienen kann. Es wurde ein erstes Experiment mit dieser Software durchgeführt, welches vielversprechende Ergebnisse geliefert hat. Es wurde gezeigt, dass die entwickelte Software stabil läuft und dass die verwendeten Sensoren Rückschlüsse auf die Aktivität und den Aufenthaltsort eines Mobiltelephonbenutzers zulassen.

Ausgehend von dieser Arbeit sind nun verschiedene Folgearbeiten und Experimente denkbar. Einerseits könnte ein Experiment durchgeführt werden, welches das in dieser Arbeit durchgeführte Experiment mit mehreren Personen über einen längeren Zeitraum fortsetzt. Ein weiteres Thema, welches nun angegangen werden könnte, ist ein Experiment, welches die Unterbrechbarkeit der Benutzer untersucht.

Zum Zeitpunkt der Abgabe dieser Diplomarbeit stehen bereits zwei konkrete Folgearbeiten fest, welche mit der in dieser Arbeit entwickelten Software durchgeführt werden sollen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Aha and Kibler, 1991] Aha, D. and Kibler, D. (1991). *Machine Learning*, chapter Instance-based learning algorithms, pages 37–66. San Mateo, CA, 6. edition.

[Devijver and Kittler, 1982] Devijver, P. and Kittler, J. (1982). *Pattern Recognition: A Statistical Approach*. Prentice-Hall International.

[Dey and Abowd, 1999] Dey, A. K. and Abowd, G. D. (1999). Towards a better understanding of context and context-awareness. Technical report, Georgia Institute of Technology.

[Egger, 2004] Egger, P. (2004). Unterbrechbarkeitsbestimmung für wearable computing applikation. Master's thesis, Universität Zürich, Institut für Informatik.

[Farringdon et al., 1999] Farringdon, J., Moore, A. J., Tilbury, N., Church, J., and Biemond, P. D. (1999). Wearable sensor badge and sensor jacket for context awareness. *The Third International Symposium onWearable Computers, 1999. Digest of Papers.*, pages 107–113.

[Gellersen et al., 2002] Gellersen, H. W., Schmidt, A., and Beigl, M. (2002). Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7:341–351.

[Hudson et al., 2003] Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., and Yang, J., editors (2003). *Predicting human interruptibility with sensors: a Wizard of Oz feasibility study*.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.

[Mühlenbrock et al., 2004] Mühlenbrock, M., Brdiczka, O., Snowdon, D., and Meunier, J.-L., editors (2004). *Learning to detect user activity and availability from a variety of sensor data*.

[Quinlan, 1993] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

[Whitaker and Kay, 2005] Whitaker, R. and Kay, J. (2005). Location and activity modelling in intelligent environments. Technical report, School of Information Technologies, University of Sydney.

A

TAPI Messages

Untenstehend finden sich die von der TAPI ausgegebenen Messages für verschiedene Ereignisse.

Eingehender Anruf	**Verpasster Anruf**

`LINE_CALLSTATE:OFFERING`
`LINE_CALLINFO:ORIGIN`
`LINE_CALLINFO:CALLERID`
`LINE_CALLINFO:CONNECTEDID`
`...`
	`LINE_CALLSTATE:OFFERING`
	`LINE_CALLINFO:ORIGIN`
	`LINE_CALLINFO:CALLERID`
	`LINE_CALLINFO:CONNECTEDID`
	`LINE_CALLSTATE:DISCONNECTED:UNAVAIL`
	`LINE_CALLSTATE:IDLE`
	`...`

Angenommener Anruf	**Beendeter Anruf**

`LINE_CALLSTATE:OFFERING`
`LINE_CALLINFO:ORIGIN`
`LINE_CALLINFO:CALLERID`
`LINE_CALLINFO:MEDIAMODE`
`LINE_CALLINFO:CONNECTEDID`
`LINE_CALLSTATE:CONNECTED`
`LINE_REPLY:5600`
`...`
	`LINE_CALLSTATE:OFFERING`
	`LINE_CALLINFO:ORIGIN`
	`LINE_CALLINFO:CALLERID`
	`LINE_CALLINFO:MEDIAMODE`
	`LINE_CALLINFO:CONNECTEDID`
	`LINE_CALLSTATE:CONNECTED`
	`LINE_REPLY:5600`
	`...`
	`LINE_CALLSTATE:DISCONNECTED:UNAVAIL`
	`LINE_CALLSTATE:IDLE`

Tabelle A.1: TAPI Messages

B

Software Settings

Nachfolgend findet sich der Inhalt der Datei *settings.txt*. Diese Datei beinhaltet sämtliche Einstellungsmöglichkeiten für die Software. Sie befindet sich im Hauptverzeichnis der Applikation. Die entsprechenden Einstellungen werden dabei im Format *Schlüssel=Wert* gespeichert. Linien, welche mit **#** oder **/** beginnen werden dabei als Kommentare angesehen und vom Parser ignoriert.

Werden Änderungen an den Einstellungen vorgenommen, muss die Software zwingend beendet und neu gestartet werden.

```
#Global settings
//where the collected data is temporarily saved
application.rootFolder=\My Documents\WMCADT
//where to move the collected data to
application.outputFolder=\Storage Card\WMCADT
//show/hide main form
application.silentMode=false
//signalling mode
//0=MUTE,1=VIBRA,2=AUDIO,3=BOTH,4=AUTO
application.signalMode=1
//maximum time to wait (in seconds) for user input, before we close all open forms
//and continue to collect data
application.maxUserResponseTime=30

#Log settings
//0=DEBUG,1=INFO,2=WARNING,3=ERROR,4=FATAL
application.log.level=2
//Where to save the log files relative to application.rootFolder
application.log.path=\log\
//prefix for log file
application.log.prefix=log_

#Serial settings
serial.on=true
//input buffer for serial
serial.bufferSize=12288
//sleep time for serial thread to catch up if no line is retrieved
serial.sleepTime=10
//Where to save serial data files relative to application.rootFolder
serial.path=\serial\
//serial filename prefix
serial.prefix=serialdata
```

```
//max elements in ring buffer (this is in Memory!)
serial.maxElements=10000
//baud rate for serial
serial.baudRate=9600
//how many milliseconds to wait until a timeout occurs for a single readline call
serial.readTimeout=1
//how many seconds to wait if a timeout occured, before we notify the user to take action
serial.maxDataTimeout=5

#Sensor settings
//start commands to send to serial
sensor.commands.start=T,A,G,M,T,G,A,M,X
//stop commands to send to serial
sensor.commands.stop=t,a,g,m,x
// Flag to define if commands should be repeated while Thread is running.
// If true, the commands will be sent to serial device in the same interval
// as the reset command.
sensor.repeatCommands=true
//commands to send to serial in case of an event start (e.g. incoming call)
sensor.commands.event.start=N
//commands to send to serial in case of an event end (e.g. call accepted/missed)
sensor.commands.event.stop=n
//interval in seconds after which to reset all sensors
sensor.reset.interval=60

#GPS settings
gps.on=true

#Picture settings
photo.on=false
//Where to save captured images relative to application.rootFolder
photo.path=\pictures\
//prefix for image names
photo.prefix=pic
//sleep time between taking pictures
photo.sleepTime=15000
//Picture resolution
photo.resolution.height=120
photo.resolution.width=160
//how may pictures should be taken
photo.count=15
//only take snapshot(s) in case of an event?
photo.snapshot=true

#Audio settings
audio.on=true
//Where to save audio streams relative to application.rootFolder
audio.path=\audio\
//prefix for audio file names
audio.prefix=stream
//how many seconds per stream
audio.stream.length=30
//how many streams should be recorded
audio.stream.count=10

#Simulator settings
simulator.on=true
//interval to raise a simulated call in seconds
simulator.interval=300
//milliseconds to collect data (i.e. inform data threads of event) before the real event is raised
simulator.event.waitForData=30000
//how long a simulated event lasts in seconds
simulator.event.maxDuration=5

#SD Card settings
//if free space percentage on SD Card drops below this value,
//a warning will be displayed
sdcard.alertFreeSpace=10
//if free space percentage on SD Card drops below this value,
//system will no longer copy the collected data.
sdcard.minFreePercentage=4
```

C

Scripts

Für die Aufbereitung der von der Software erfassten und gespeicherten Rohdaten wurden einige Scripts geschrieben, um die häufigsten Arbeitsschritte zu automatisieren. Damit diese Scripts für Folgearbeiten wiederverwendet werden können, soll nachfolgend eine kurze Anleitung gegeben werden. Die Scripts finden sich auf der beigelegten CD-ROM im Verzeichnis */Scripts*.

C.1 Importieren der Rohdaten in Matlab

Die Daten von der Speicherkarte des Mobilephones können in der bestehenden Ordnerstruktur verwendet werden. Nach dem transferieren des Inhalts des Ordners */WMCADT* von der Speicherkarte auf den PC können die Rohdaten mittels dem MatLab-Script *import_event_data.m* in MatLab importiert werden. Der Funktionsaufruf zum Importieren der Daten lautet wie folgt:

```
import_event_data('C:/pfad/zu/den/rohdaten')
```

Dieser Aufruf erstellt im übergebenen Verzeichnis einen neuen Ordner */matlab* und generiert für jeden aufgezeichneten Event ein MatLab Datenfile (.mat). Eine solche Datei beinhaltet nun ein MatLab-Struct, welches sämtliche Daten beinhaltet.

C.2 Generieren der ARFF-Files für Weka

Nachdem die Rohdaten importiert wurden, können sie mittels dem MatLab-Script *serial2ARFF.m* in das von Weka benötigte ARFF-Format gebracht werden. Der Funktionsaufruf dieses Scripts sieht dabei wie folgt aus:

```
serial2ARFF(event_directory, max_interval_mins, desired_interval_min,
                                filename)
```

Die Funktionsparameter sind die folgenden:

- *event_directory*:
  Speicherort der im vorherigen Schritt generierten .mat-Files
- *max_interval_mins*:
  Maximale Aufnahmelänge der Daten vor einem Event (in Minuten).
- *desired_interval_min*:
  Intervalllänge, für welche die Features erstellt werden sollen (in Minuten).
- *filename*:
  Name der zu erstellenden ARFF-Datei. Die Datei wird im Verzeichnis, welches als Parameter bei *event_directory* angegeben wurde, im (allenfalls neu erstellten) Unterordner */jam* abgelegt.

Ein Beispielaufruf:

```
serial2ARFF('C:/pfad/zu/den/rohdaten/matlab', 5, 1, 'myFilename.arff')
```

Falls für die gleichen Daten mehrere Intervalle untersucht werden sollen, kann das MatLab-Script *generateARFFs.m* verwendete werden. Die Funktion weist die folgende Signatur auf:

```
generateARFFs(input_dir, max_length, intervals, output_file_prefix)
```

Die Funktionsparameter sind die folgenden:

- *input_dir*:
  Speicherort der im vorherigen Schritt generierten .mat-Files
- *max_length*:
  Maximale Aufnahmelänge der Daten vor einem Event (in Minuten).

- *intervals*:
  Matrix mit Intervalllängen, für welche die Features erstellt werden sollen (in Minuten).

- *output_file_prefix*:
  Präfix der zu erstellenden ARFF-Dateien. Der Name der erstellten Dateien setzt sich zusammen aus diesem Präfix und der jeweils verwendeten Intervalllänge in Sekunden. Die Dateien werden im Verzeichnis, welches als Parameter bei *input_dir* angegeben wurde, im (allenfalls neu erstellten) Unterordner */jam* abgelegt.

Ein Beispielaufruf:

```
generateARFFs('C:/pfad/zu/den/rohdaten/matlab',5,[1/12,0.75,13,4,5],
                                '080507')
```

C.3 Data-Mining mit Weka über die Windows Command Line

Weka kann entweder durch das Java-GUI bedient, oder per Aufruf aus der Kommandozeile verwendet werden. Sollen mehrere ARFF-Dateien mit derselben Featurekombination unersucht werden, bietet sich die Verwendung der sehr einfachen Batch-Dateien im Ordner */scripts/batch* an. Um dieses Script zu verwenden, müssen die ARFF-Dateien, welche untersucht werden sollen, in das Unterverzeichnis */data* kopiert werden. Danach kann der Batch-Vorgang folgendermassen ausgelöst werden (Aufruf über Kommandozeile mit aktuellem Arbeitsverzeichnis */scripts/batch*:

```
weka_it ‘‘Zu entfernende Features’’
```

“Zu entfernende Features” bezeichnet dabei die Features, welche von Weka vor der Auswertung aus der Kombination entfernt werden sollen. Der Parameter “1-3,27” würde beispielsweise die Features 1 bis 3 (inklusive 1 & 3) sowie das 27. Feature entfernen.
Ein Beispielaufruf:

```
weka_it ‘‘1-3,27’’
```

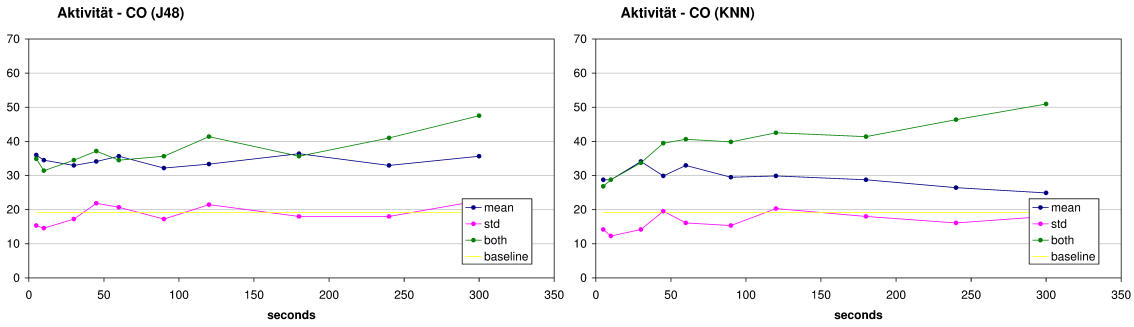
Das Ergebnis des Batch-Vorganges wird im Unterverzeichnis */output* abgelegt. Anmerkung: Die darin enthaltenen Dateien werden ohne Warnung überschrieben, falls mehrere Durchläufe mit den selben ARFF-Dateien gemacht werden.

Ob der J48 oder der Ibk Algorithmus verwendet wird, lässt sich durch eine kleine Änderung in der Batch-Datei *weka_it.bat* festlegen (*weka_single_J48* für J48, *weka_single_NN* für Ibk). Allfällige Parameter für die Algorithmen können ebenfalls in den Batch-Dateien *weka_single_J48.bat* für J48 respektive *weka_single_NN.bat* für Ibk konfiguriert werden.

D

Weitere Auswertungen

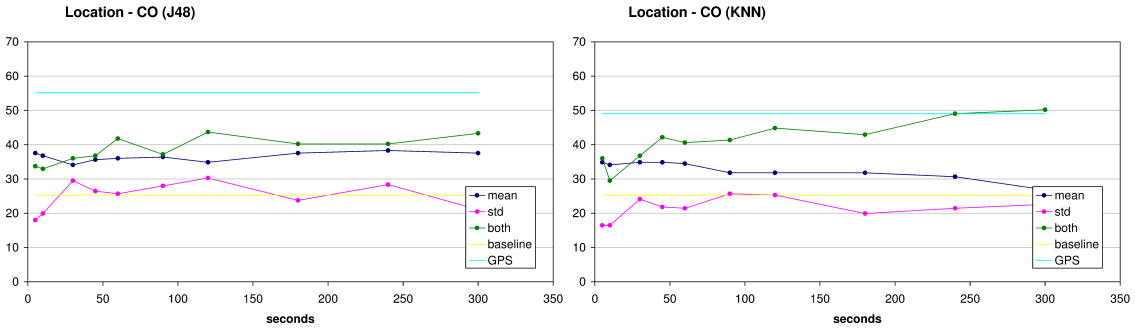
Da die von den einzelnen Gas-Sensoren erzielten Werte durch eine Kombination derselben deutlich verbessert werden konnten, wurden diese beiden Sensoren zusammengefasst. Nachfolgend sollen die Ergebnisse der einzelnen Gas-Sensoren dennoch aufgeführt werden.



(a) J48

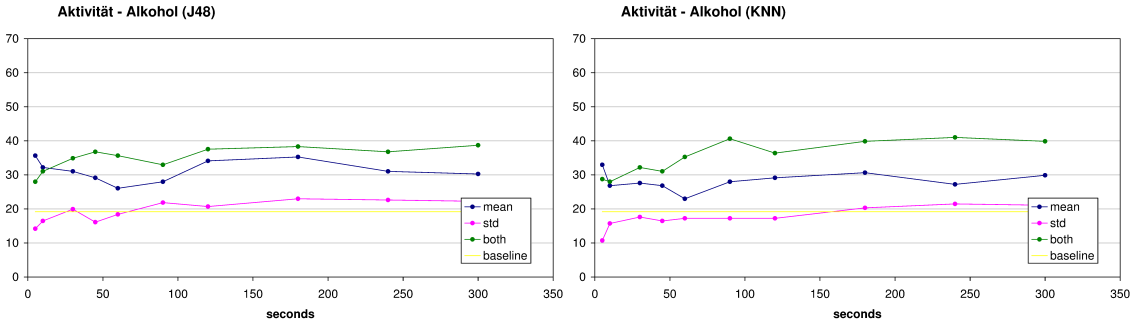
(b) KNN

Abbildung D.1: Der Prior wird nur vom Durchschnitt und der Kombination von Durchschnitt und Standardabweichung übertroffen. Interessant erscheint die deutliche Verbesserung, welche eine Kombination der beiden Features beim KNN Algorithmus ergibt.



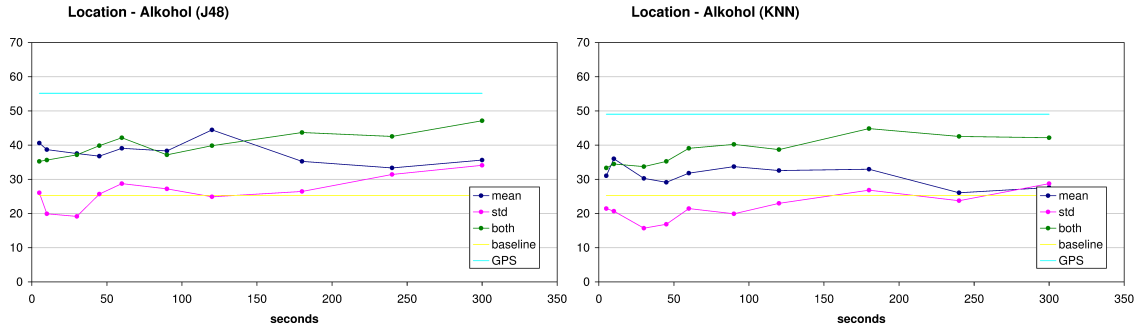
(a) J48 (b) KNN

Abbildung D.2: Der Prior wird auch für den Aufenthaltsort nur vom Durchschnitt und der Kombination übertroffen. Wie bereits bei der Aktivität ergibt die Kombination beider Features beim KNN Algorithmus eine sehr deutliche Verbesserung der Voraussagequalität. Ab einer Intervalllänge von 240 Sekunden wird sogar der GPS-Benchmark erreicht bzw. leicht übertroffen.



(a) J48 (b) KNN

Abbildung D.3: Dieser Sensor zeigt ein gegenüber dem CO-Gas Sensor ein sehr ähnliches Bild. Die Standardabweichung vermag für sich alleine genommen, im Gegensatz zum Durchschnitt und der Kombination beider Features, den Prior wiederum nicht deutlich zu schlagen. Die vom CO-Gas Sensor erzielten Voraussagewerte werden jedoch weder vom Durchschnitt, noch von der Kombination erreicht.



(a) J48 (b) KNN

Abbildung D.4: Auch für den Aufenthaltsort zeigt sich ein sehr ähnliches Bild. Der Durchschnitt und die Kombination beider Features erzielen wiederum Werte, welche über dem Prior liegen, während die Standardabweichung für sich alleine genommen dies nicht vermag. Auch hier werden die vom CO-Gas Sensor erzielten Werte nicht ganz erreicht.

E

CD

Dieser Arbeit ist eine CD-Rom beigelegt. Sie beinhaltet diese Arbeit in elektronischer Form (als PDF), die Resultate des durchgeführten Experiments, die verwendeten Scripts sowie sämtlichen Quellcode der in dieser Arbeit vorgestellten Software. Zusätzlich liegen der Arbeit zwei DVDs bei, welche die während dem Experiment gesammelten Daten beinhalten.

Die Tabellen E.1 & E.2 zeigen eine Übersicht über den Inhalt des jeweiligen Datenträgers.

Verzeichnis	**Inhalt**
Diplomarbeit	Diplomarbeit als PDF
Zusammenfassung	Zusammenfassung (Deutsch & Englisch) der Diplomarbeit als PDF
Resultate	Resultate des Experiments
LaTeX	LaTeX- Dateien dieser Diplomarbeit
Scripts	Die Scripts, welche in dieser Arbeit zu Einsatz kamen
Software	Quellcode der Software

Tabelle E.1: Inhalt CD-ROM

Verzeichnis	**Inhalt**
Rohdaten	Gesammelte Rohdaten des Experiments
MatLab	Rohdaten im MatLab Format (.mat)
ARFF	Für Weka aufbereitete Daten im ARFF Format

Tabelle E.2: Inhalt DVDs