Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl 12 Prof. Dr.-Ing. Gernot A. Fink

Endbericht

CampusGuide - Ein mobiler Roboterassistent als Orientierungshilfe unter Verwendung webbasierter Services

Endbericht von:
Nazli Bas, Igor Blyufshteyn, Steve Brischke, Andreas Mülle, Andreas Niemöller, Jörg Nitschke, Jan Nörtemann, Stefan Ostwald, Dominik Thalmann, Henning Timm, Tian Zhang

Betreuer:
Prof. Dr.-Ing. Gernot A. Fink
Dipl.-Ing. Jan Richarz
Dipl.-Inf. Marius Hennecke

Inhaltsverzeichnis

1 Einführung **4**
- 1.1 Der mobile Roboter auf dem Campus . . . 5
- 1.2 Anforderungen und Zielsetzung . . . 5
- 1.3 Aufbau des Endberichts . . . 8

2 Verwandte Arbeiten **9**
- 2.1 Ähnliche Projekte . . . 9
- 2.2 Navigation . . . 12
- 2.3 Mensch Maschine Interaktion . . . 15
  - 2.3.1 Personendetektion . . . 15
  - 2.3.2 Emotionen bei Robotern . . . 17

3 Grundlagen **19**
- 3.1 Hardware . . . 19
  - 3.1.1 Roboterplattform . . . 21
  - 3.1.2 Sensoren . . . 22
  - 3.1.3 Smartphone . . . 23
- 3.2 Player/Stage . . . 24

4 Realisierung **27**
- 4.1 Die Softwarearchitektur . . . 27
- 4.2 Kommunikationsarchitektur zwischen Nutzer, Smartphone und Roboter . . . 30
  - 4.2.1 Einführung in das Android-Betriebssystem . . . 30
    - 4.2.1.1 Intents . . . 31
  - 4.2.2 Bluetooth . . . 31
  - 4.2.3 Nutzerfindung per SMS . . . 32
  - 4.2.4 GPS . . . 32
  - 4.2.5 Bluetooth-Abstandsmessung . . . 33
  - 4.2.6 Kommunikation mit dem Roboter . . . 34
- 4.3 Navigation . . . 37

4.3.1 Lokalisierung . . . 37
4.3.2 Kartenmaterial der OpenStreetMap . . . 39
4.3.2.1 OSM-Daten automatisch herunterladen . . . 39
4.3.2.2 GPS-Wegpunkte extrahieren . . . 40
4.3.3 Pfadplanung . . . 40
4.3.4 Hindernisvermeidung . . . 42
4.4 Befahrbarkeit . . . 43
4.4.1 Pipeline . . . 44
4.4.2 Farberkenner . . . 45
4.4.3 Annotation . . . 45
4.4.4 Merkmalsextraktion . . . 46
4.4.4.1 Farbhistogramme . . . 49
4.4.4.2 Gabor . . . 49
4.4.5 Klassifikation . . . 51
4.5 Mensch-Maschine-Interaktion . . . 52
4.5.1 Personendetektion . . . 52
4.5.1.1 Ablauf der Personendetektion . . . 52
4.5.1.2 Beinpaardetektor . . . 54
4.5.1.3 Gesichtsdetektor . . . 58
4.5.2 Benutzerschnittstelle . . . 60
4.5.2.1 Interaktion mit einem Benutzer . . . 60
4.5.2.2 Interaktion mit einem Administrator . . . 62
4.5.3 Emotionen . . . 62
4.6 Webserver . . . 66

5 Evaluierung 68
5.1 Bluetooth-Abstandsmessung . . . 68
5.2 Befahrbarkeit . . . 70
5.2.1 Datenaufnahme . . . 70
5.2.2 Auswahl der Merkmale . . . 72
5.2.3 Training der neuronalen Netze . . . 73
5.2.4 Ergebnisse . . . 74
5.3 Personendetektion . . . 78
5.3.1 Beinpaardetektor . . . 79
5.3.2 Personendetektor . . . 85
5.4 Gesamtsystem . . . 93
5.4.1 Aufbau und Durchführung . . . 93
5.4.2 Ergebnisse . . . 95

6 Zusammenfassung und Ausblick 98

A Vollständige Liste der Zustandscodes **100**

B Bilder **104**
B.1 GUI 105
B.2 Evaluation der Bluetooth-Abstandsmessung 112
B.3 Evaluation des Beinpaardetektors 115
B.4 Evaluation des Personendetektors 117

C Benutzerstudie **120**

Kapitel 1

Einführung

Die Entwicklung von selbstständig agierenden Robotern stellt aktuell ein wichtiges Ziel der Robotik dar. Unter kontrollierten Umgebungsbedingungen können Industrie- und Serviceroboter bereits heute autonom Arbeitsabläufe durchführen. Beispiele hierfür sind Fertigungsstraßen in der Automobilindustrie oder Serviceroboter als Haushaltshilfe. Diese Roboter werden für einen bestimmten Arbeitsschritt entworfen und sind auf die Ausführung dieser Tätigkeit festgelegt. Zusätzlich kommen sie meist in einer bekannten oder kontrollierten Umgebung zum Einsatz.

Wesentlich herausfordernder ist es jedoch, mobile autonome Roboter zu entwickeln, die auch in unbekannten Umgebungen und unter wechselnden Bedingungen operieren können. Dies ist besonders für Roboter wichtig, die mit Menschen in Interaktion treten sollen. Insbesondere müssen diese Systeme in der Lage sein dynamisch auf unvorhergesehene Ereignisse zu reagieren.

Gerade Umgebungen mit vielen Passanten stellen für autonome Roboter eine Herausforderung dar, was die Entwicklung von Systemen für diesen Anwendungsbereich zu einem interessanten Forschungsthema macht. Bereits in den 90er Jahren wurden die ersten Tour-Guide Roboter RHINO [1] und Minerva [2] entwickelt. Sie dienten dazu, Menschen in öffentlichen Gebäuden, wie in Museen, zu begleiten und zu führen. Spätere Entwicklungen waren die Tour-Guides Jinny [3] und ShopBot [4]. Diese Roboter kamen alle in kontrollierten Umgebungen zum Einsatz. Es ist jedoch wünschenswert ein System zu entwickeln, dass nicht auf einen bestimmten Einsatzbereich festgelegt ist. Gerade ein Guide, der sowohl in Gebäuden, als auch in Außenarealen operieren kann, birgt großes Potential für viele Anwendungsbereiche.

1.1 Der mobile Roboter auf dem Campus

Eine Situation wie die Folgende werden die meisten Menschen bereits erlebt haben. Man befindet sich zu Gast auf einem großen Gelände wie z.B. einer Messehalle, einem Flughafen oder dem Campus einer Universität und sucht einen bestimmten Ort. Dabei kann es sich, je nach Situation, um einen Messestand, ein Büro oder auch nur das richtige Regal in einem Geschäft handeln. Zusätzlich drängt die Zeit: Man muss pünktlich zu einem Termin erscheinen, beziehungsweise ein Flugzeug erreichen. Gerade unter Zeitdruck ist es leicht möglich in einer unbekannten Umgebung die Orientierung zu verlieren. In einer derartigen Situation wäre es wünschenswert, einen auf Abruf verfügbaren ortskundigen Führer zu haben.

An diesem Punkt kommt CampusGuide, ein per SMS abrufbarer Roboter-Fremdenführer, ins Spiel. Er soll auf dem Gelände der TU Dortmund operieren um derartige Probleme zu lösen. Der Roboter kann jedoch nicht nur bei der Orientierung hilfreich sein, sondern auch mit seiner Fähigkeit, Informationen für den Benutzer aufzubereiten. Viele Benutzer werden sich auf der Suche nach Personen befinden. Deren Aufenthaltsort zu ermitteln kann jedoch, obwohl die Daten online vollständig vorliegen, eine Herausforderung darstellen. Auch diese Schwierigkeit sollte ein guter Fremdenführer überwinden können. Dem Roboter steht dafür, zusätzlich zu internen Datenbanken, die Möglichkeit zur Verfügung, online nach Personen oder Adressen zu suchen. Auch der Zugriff auf frei nutzbares Kartenmaterial wie z.B. das der OpenStreetMap(OSM)[1] ist möglich und kann zur Suche nach Zielen und befahrbaren Wegen verwendet werden.

1.2 Anforderungen und Zielsetzung

Um die genaue Zielsetzung für das System zu formulieren betrachten wir zuerst einen typischen Anwendungsfall von CampusGuide (siehe Abb. 1.1).

Ein Besucher befindet sich auf dem Campus der TU Dortmund und ist auf der Suche nach einer bestimmen Person oder einem Gebäude. Wenn ihm dies nicht auf Anhieb gelingt, oder er sich nicht die Mühe machen möchte zu suchen, kann er mit einer SMS den CampusGuide zu seiner momentanen Position rufen.

Der Roboter empfängt diese SMS und berechnet auf Kartendaten, die aus der OSM erzeugt wurden, einen Weg zum Standort des Benutzers und fährt diesen an. Zeitgleich erzeugt er ein Passwort, mit dem der Benutzer sich am Roboter authentifizieren kann und schickt dieses dem Benutzer per SMS zu.

[1] http://www.openstreetmap.org/

(a) Ein Gast auf dem Campus kann sein Ziel nicht finden

(b) Der Benutzer ruft den Roboter per SMS

(c) Der Roboter erreicht den Benutzer

(d) Der Benutzer wählt ein Ziel

(e) Der Roboter führt den Nutzer zum Ziel

(f) Das Ziel wurde erreicht, der Roboter beendet die Führung

Abbildung 1.1: Ein typischer Anwendungsfall

Auf diesem Weg erhält der Benutzer ebenfalls eine Bestätigung, dass sich der Roboter auf dem Weg zu ihm befindet.

Sobald CampusGuide am Zielort angekommen ist, macht er sich auf die Suche nach dem Benutzer. Dafür verwendet er eine Kamera, deren Bilder mit Hilfe einer Gesichtserkennungssoftware analysiert werden, und die Daten eines Beinpaar-Detektors.

Der Benutzer kann an den Roboter herantreten und sich, mittels des vom Roboter verschickten Passworts, bei CampusGuide anmelden. Dann steht ihm eine Benutzeroberfläche zur Verfügung, mit der er nach seinem Zielort oder der Zielperson suchen kann.

Wenn der Benutzer ein Ziel ausgewählt hat, berechnet CampusGuide, erneut auf Daten von OSM, einen Pfad zum Ziel. Während der Roboter dieses Ziel anfährt, achtet er darauf, dass er sich nicht zu weit vom Benutzer entfernt.

Sobald das Ziel erreicht ist, gibt CampusGuide dies dem Benutzer bekannt und bleibt stehen. Der Nutzer kann entweder einen neuen Zielort eingeben oder sich abmelden und den Roboter damit für andere Aufgaben wieder freigeben.

Das Ziel der Projektgruppe ist es, ein Softwareframework zu entwerfen, das diese Aufgaben erfüllen kann. Für die Realisierung der Personendetektion, der Mensch-Maschine-Interaktion und der autonomen Navigation kommen außerdem geeignete Verfahren der Mustererkennung zur Anwendung. Als Hardware stehen uns die Roboterplattform SCITOS G5[2] und das Smartphone HTC Legend [3] zur Verfügung. Aus dem Anwendungsfall ergeben sich die folgenden Anforderungen:

Die Roboterplattform soll in der Lage sein, sich selbstständig auf dem Campus zu bewegen. Dazu muss sie zum einen in der Lage sein Hindernisse zu erkennen und diesen auszuweichen. Zum anderen muss sie selbstständig, mit Hilfe einer Karte, Pfade planen und diese abfahren können. Dafür soll sowohl der GPS-Empfänger des Smartphones, als auch die eigene Sensorik des Roboters (Laser und Kamera) verwendet werden. Die Karte die für diese Navigation verwendet wird soll der Roboter automatisch aus Daten der OSM erzeugen. Um die Kommunikation mit dem Benutzer zu ermöglichen, muss das Smartphone an die Roboterplattform angebunden werden. Dieses soll der Software auf dem Roboter außerdem Zugriff auf GPS Daten, Internetanbindung, Bluetooth- und Telefonfunktionen des Smartphones erlauben. Über diese Verbindung soll das selbstständige Holen von Daten aus dem Internet, insbesondere von Kartendaten der OSM, sowie das Bereitstellen von

[2] http://www.metralabs.com

[3] http://www.htc.com/de/product/legend/overview.html

Daten für eine Webapplikation zum Verfolgen der Position des Roboters realisiert werden. Zusätzlich soll der Roboter in der Lage sein Menschen, und somit mögliche Interaktionspartner, zu identifizieren. Dazu können der Laser-Entfernungsmesser und eine Kamera verwendet werden. Des Weiteren muss ein Konzept zur Interaktion mit dem Benutzer, sowohl per SMS, als auch mit dem Touchscreen des Roboters, entwickelt werden.

1.3 Aufbau des Endberichts

Um einen Überblick über ähnliche Projekte zu geben, werden in Kapitel zwei zunächst verwandte Arbeiten vorgestellt. Darüberhinaus werden dort einige Arbeiten vorgestellt, auf deren Grundlage wir Lösungen für die in Kap. 1.2 identifizierten Probleme entwickelt haben. In Kapitel drei werden Grundlagen, wie zum Beispiel die von uns verwendete Hardware, vorgestellt. Insbesondere wird dort auf die Sensorik des Roboters, die Software des Smartphones und die Software zur Steuerung des Roboters eingegangen. Kapitel vier enthält die Beschreibung des von uns entwickelten Systems. Wir betrachten dort den allgemeinen Aufbau des Frameworks sowie die Realisierung der einzelnen Module. Kapitel fünf beschreibt die Evaluation der einzelnen Komponenten sowie den Benutzertest des Gesamtsystems. Abschließend werden in Kapitel sechs die Ergebnisse der Projektgruppe zusammengefasst und Ausblicke auf mögliche Erweiterungen und Verbesserungen des Systems gegeben.

Kapitel 2

Verwandte Arbeiten

Die autonome Robotik ist bereits seit geraumer Zeit ein aktives Forschungsfeld. Daher existiert eine Vielzahl von Arbeiten, die sich mit Themen beschäftigen, die für die Projektgruppe CampusGuide wichtig sind. Im Folgenden wird ein Überblick über wissenschaftliche Arbeiten und Publikationen gegeben, die sich mit verwandten Problemen und den Grundlagen für unser System beschäftigen.

2.1 Ähnliche Projekte

Für die Projektgruppe haben wissenschaftliche Arbeiten, mit einer ähnlichen Zielsetzung, eine besondere Bedeutung. Dies sind insbesondere *Tourguides* die in verschiedenen Umgebungen Personen zu ihren Zielorten führen sollen.

In [4] wird der Service-Roboter ShopBot vorgestellt, der in einem Baumarkt als Guide fungiert. Für diese Aufgabe benötigt der Roboter zwei wichtige Fertigkeiten. Er muss selbstständig in dem Geschäft navigieren sowie Personen erkennen und ihre Position verfolgen können.

Bei der Navigation werden zwei verschiedene Ansätze getestet. Der erste Ansatz nutzt einen *Simultaneous Localization and Mapping* (SLAM)-Algorithmus [5], um aus den Sonar- und Laser-Messungen des Roboters eine Karte des Geschäfts zu erstellen. Dabei wird nach Übereinstimmungen einer lokalen mit einer globalen Karte gesucht, um die Position des Roboters zu ermitteln. Der zweite Ansatz verfolgt das Ziel, Kamerabilder von der Umgebung des Roboters als Navigationsgrundlage zu nutzen. Dazu wurden im gesamten Geschäft, in Abständen von 20 cm, 360° Aufnahmen der Umgebung gemacht. Von diesen Bildern wurden Histogramme im HSV-Farbraum erstellt und den entsprechenden Positionen und Richtungen zugeordnet. Durch die Beziehung mehrerer aufeinanderfolgender Aufnahmen erhält man Informationen durch

den geometrischen Unterschied der Aufnahmen, durch die veränderte Position, und den visuellen Unterschied. Der visuelle Unterschied hängt stark von der Entfernung zum Hindernis ab, sodass bei nahen Objekten die Unterschiede zwischen den Aufnahmen größer sind als bei weit entfernten Objekten. So entsteht ein Graph, dessen Knoten verschiedene Positionen und Blickrichtungen repräsentieren und die extrahierten Merkmale der Umgebung abspeichern. Während der Navigation im Geschäft sucht der Roboter im Graphen den Knoten, der die größte Ähnlichkeit mit den Merkmalen der aktuellen Umgebung hat und bestimmt damit die eigene Position und Richtung. Die Tests des ShopBots zeigten, dass der erste Ansatz leicht bessere Ergebnisse lieferte.

Die Personenerkennung des ShopBots wird durch eine Kombination von drei Methoden erreicht. Die erste Methode wendet einen Beinpaardetektor auf die Entfernungsmessungen eines horizontalen 2D-Laserscans an. Bei der zweiten Methode werden die Kamera-Bilder nach Hautfarben abgesucht. Die dritte Methode nutzt einen Viola-Jones-Gesichtserkenner [6]. Wenn alle drei Methoden eine gemeinsame Detektion aufweisen, wurde eine Person vermutet.

Die Tests zeigen, dass sich der Roboter in der geschlossenen Umgebung des Baumarktes gut zurechtfindet und die Personen in den meisten Fällen zum gewünschten Ziel geleiten kann. Die Personendetektion ist in der Lage die meisten Personen korrekt zu detektieren und während der Route zum Zielort korrekt zu verfolgen.

Ein weiteres Szenario, in dem Roboter als Führer bereits eingesetzt und ausführlich getestet wurden, sind *Tourguides* in Museen [7][2][3]. Dabei soll der Roboter selbstständig durch das Museum navigieren, Führungen durch das Museum machen und den Besuchern Informationen zu den Ausstellungsstücken geben. Die größten Probleme entstehen dadurch, dass der Roboter oft in Bereichen fahren muss, in denen sich viele Menschen aufhalten. Diese dürfen nicht gefährdet oder behindert werden. Trotzdem muss der Roboter auch mit den Personen interagieren, die ihn nutzen wollen. Außerdem können meist keine Modifikationen an der Umgebung vorgenommen werden, um dem Roboter die Navigation zu erleichtern.

Im Deutschen Museum in Bonn wurde der Roboter RHINO [7] eingesetzt und getestet. Dieser Roboter kann seine Umgebung mit Laser-Scannern, Sonar- und Infrarot-Sensoren erfassen. Für die Navigation des Roboters wurde eine Karte des Museums erstellt, auf der alle Hindernisse eingezeichnet sind. Insbesondere sind auch feste Hindernisse eingetragen, die von den Sensoren nicht erfasst werden können, wie Glasscheiben. Beim Einsatz im Museum trifft der Roboter nicht nur auf feste, sondern auch auf bewegliche Hindernisse, insbesondere Personen. Erkennt der Roboter ein Hindernis oder

eine freie Fläche, so wird auf einer lokalen Karte ein entsprechender Wahrscheinlichkeitwert für ein Hindernis eingetragen. Diese Wahrscheinlichkeiten werden im Laufe der Zeit angepasst, um zu modellieren, dass sich die Umgebung dynamisch ändern kann. Damit der Roboter sich lokalisieren kann, vergleicht er die Sensormessungen mit der Umgebungskarte, um daraus die wahrscheinlichsten Positionen des Roboters zu ermitteln. Dabei besteht das Problem, dass Personen in der Umgebung des Roboters stehen können und somit die Sensoren blockieren. Dadurch kann der Roboter die eingezeichneten Hindernisse auf der Karte nicht erkennen. Um dieses Problem auszugleichen, wurde ein modifizierter Markov Lokalisierungsalgorithmus [8] benutzt, der die Messungen in zwei Kategorien einteilt. Wenn die Messungen nicht mit den erwarteten Hindernissen aus der Karte übereinstimmen, werden die Messungen nicht für die Lokalisierung des Roboters verwendet. Beim Einsatz im Museum konnte RHINO in fast allen Fällen Personen zuverlässig zu ihrem Zielort führen. Allerdings stieß RHINO im Einsatz immer wieder auf das Problem, dass Personen absichtlich den Weg blockierten, wodurch es erheblich erschwert wurde, einen Weg zum Ziel zu finden.

Eine Weiterentwicklung von RHINO ist der Roboter Minerva [2], der im Smithsonian's National Museum of American History zum Einsatz kam. Hierbei wurde die Navigation des Roboters dadurch verbessert, dass zusätzlich zu den bisherigen Sensormessungen noch Kamerabilder von der Decke aufgenommen werden. Anhand der Struktur der Decke versucht der Roboter seine Position und Ausrichtung zu bestimmen, falls die restlichen Sensordaten nicht ausreichend sind. Dies ist insbesondere nötig, wenn sich Minerva auf einer großen freien Fläche ohne sonstige Lokalisierungspunkte befindet. Das Problem, dass Personen den Roboter absichtlich blockieren, konnte bei Minerva zum größten Teil gelöst werden, da er ein künstliches Gesicht hat, mit dem er Emotionen darstellen kann. Durch die Darstellung von Emotionen empfinden Personen den Roboter deutlich menschenähnlicher und blockieren wesentlich seltener seinen Weg. So entstehen nicht so oft Situationen, in denen Minerva keinen Pfad zum Ziel findet, weil Personen im Weg stehen.

Der Roboter Jinny [3] ist am KIST (Korea Institute of Science and Technology) im Einsatz. Dieser verwendet für die Navigation zwei Laser-Scanner und zwei Infrarot-Sensoren. Jinny erweitert die bestehenden Ansätze dadurch, dass er sein Navigationsverhalten der aktuellen Situation anpasst. Dabei nutzt er insgesamt vier Verhaltensweisen. Eine allgemeine Verhaltensweise, bei der der Roboter versucht möglichst direkt auf das Ziel zuzufahren und dabei Hindernissen auszuweichen, eine Verhaltensweise, die dem Verlauf von Wänden folgt, eine Variante die Personen auffordert aus dem Weg zu gehen, wenn der Roboter von vielen Personen umringt ist, und eine Variante, in der der Roboter ferngesteuert werden kann. Durch dieses angepasste

Verhalten kann Jinny z.B. schneller durch Menschenmengen navigieren, da er nicht von vornherein versucht allen Personen auszuweichen, sondern sie per Sprachausgabe auffordert, ihm Platz zu machen.

2.2 Navigation

Beim Einsatz in Außenbereichen treffen Roboter auf verschiedene Untergründe, die nicht alle für den Roboter befahrbar sind. Es müssen dabei Methoden gefunden werden, die diese Untergründe erkennen und verhindern, dass der Roboter auf diesen Untergründen fährt. In der Vergangenheit wurden verschiedene Ansätze getestet und bei einigen Wettbewerben miteinander verglichen.

Die *DARPA Grand Challenge* hat zu großen Fortschritten in dem Bereich der autonomen Roboter geführt. Dieser Wettbewerb wurde von der *Defense Advanced Research Projects Agency* ausgerufen. Dabei bestand die Aufgabe darin, ein Fahrzeug zu entwickeln, dass eine Strecke von 150 Meilen durch die Mojavewüste selbstständig, in möglichst geringer Zeit, zurücklegt. Auf dieser Strecke muss der Roboter auf verschiedenstem Terrain fahren und Hindernissen auf der Strecke selbstständig ausweichen. Die Strecke besteht aus Abschnitten mit gepflasterten, bis hin zu Abschnitten mit unbefestigtem Terrain. Dadurch ergeben sich verschiedene Anforderungen an die Fahrzeuge und an die Lösungsansätze zur Vermeidung von Hindernissen. So müssen zum Beispiel Felsen am Wegesrand oder zu große Steigungen erkannt und vermieden werden.

In [9], [10] und [11] werden drei Fahrzeuge beschrieben, die an der *DARPA Grand Challenge* 2004 bzw. 2005 teilgenommen haben. Diese drei Fahrzeuge nutzen verschiedene Techniken, um die Umgebung um das Fahrzeug herum zu erkennen und aus diesen Daten Hindernisse abzuleiten. Der in [9] beschriebene CajunBot nutzt zwei Laser-Entfernungsmesser, die schräg auf den Boden vor dem Fahrzeug gerichtet sind, kombiniert mit einem Trägheitsnavigationssystem. Diese Kombination ermöglicht es dem Fahrzeug, die Änderungen der Entfernungsdaten, die durch Bodenwellen entstehen, auszugleichen und so falsch erkannte Hindernisse zu entfernen.

Der Gewinner des Wettbewerbes 2005 wird in [10] beschrieben. Der Roboter Stanley nutzt insgesamt fünf Laser-Entfernungsmesser und eine Farbkamera. Die Laserscanner sind in verschiedenen Winkeln auf den Bereich vor dem Fahrzeug ausgerichtet, um Hindernisse in einem Bereich bis zu einer Entfernung von 25m zu erkennen. Da diese Entfernung aber nicht ausreicht, um das Fahrzeug mit ausreichender Geschwindigkeit zu fahren, wird für die Erkennung weit entfernter Hindernisse die Kamera genutzt. Die Kamera ist

so ausgerichtet, dass sie sowohl den Nahbereich vor dem Fahrzeug abdeckt, als auch weiter entfernte Punkte. Auf den Kamerabildern wird dabei nach Bereichen gesucht, die von den Laser-Entfernungsmessern als befahrbar klassifiziert wurden. Weit entfernte Bereiche auf den Kamerabildern, die dieselben optischen Eigenschaften haben wie die bereits als befahrbar klassifizierten Bereiche, werden ebenfalls als befahrbar angenommen. Dies erlaubt dem Fahrzeug mit einer höheren Geschwindigkeit zu fahren, da ansonsten die Geschwindigkeit, in Abhängigkeit zum Bremsweg, an die maximale Reichweite der Laser-Entfernungsmesser angepasst werden müsste.

Das dritte Fahrzeug [11], nutzt eine Kombination mehrerer unterschiedlicher Sensoren. Zwei Radar-Sensoren erkennen Hindernisse bereits in großer Entfernung. Der Bereich der mittleren Entfernung zum Fahrzeug wird von insgesamt vier Laser-Entfernungsmessern und der Nahbereich von zwölf Ultraschall-Sensoren abgetastet. Die unterschiedlichen Reichweiten und Genauigkeiten der Sensoren bieten die Möglichkeit, in verschiedenen Distanzen Hindernisse erfolgreich zu erkennen. Unterstützt wird dieses System von zwei Farbkameras. Auf den Kamerabildern wird zum einen nach Kanten gesucht, um Hindernisse zu erkennen. Außerdem wird davon ausgegangen, dass sich befahrbare Bereiche optisch deutlich von den nicht-befahrbaren Bereichen unterscheiden.

Diese Fahrzeuge bieten verschiedene erfolgreiche Ansätze für die Navigation in Außenbereichen. Der große Unterschied zu unserem Szenario besteht allerdings darin, dass die Fahrzeuge, im Gegensatz zu unserem Roboter, von vornherein für Außenbereiche konzipiert wurden. Dadurch sind sie in der Lage über unbefestigtes Terrain zu fahren, sodass sie nur großen Hindernissen ausweichen müssen. Außerdem nutzen alle drei Fahrzeuge mehrere Laser-Entfernungsmesser, um die Umgebung in mehreren Entfernungen nach Hindernissen abzusuchen. Ein solches Vorgehen kommt für uns aber aus Kostengründen nicht in Frage.

Ein weiterer Wettbewerb, der großen Einfluss auf die Entwicklung autonomer Roboter hatte, war die *Tsukuba Challenge* 2007. Bei diesem Wettbewerb bestand die Aufgabe darin, einen Roboter zu entwickeln, der sich auf einem realen Fußweg autonom und ohne zusätzliche Markierungen des Weges, bewegen soll. Die Strecke ist insgesamt ein Kilometer lang und hat mehrere unterschiedliche Untergründe, die teilweise auch mit Laub bedeckt sind. Auf dem Weg soll der Roboter Fußgängern und Fahrradfahrern ausweichen, ohne diese stark zu behindern. Morales et al. beschreiben in [12] den Sieger dieses Wettbewerbs. Der Roboter nutzt insgesamt vier Laser-Entfernungsmesser, um Hindernisse und den Untergrund zu erkennen. Davon sind zwei waagerecht ausgerichtet und suchen nach Hindernissen in der Nähe des Roboters. Die anderen beiden Scanner erfassen in unterschiedlichen Winkeln den Be-

reich vor dem Roboter, um den Boden zu erfassen. Der Roboter versucht, vorher festgelegte GPS-Punkte anzufahren und auf dem Weg dorthin Hindernissen und unpassierbarem Untergrund auszuweichen. Dabei wird jeder Untergrund, der von den Laser-Scannern als flach erkannt wird, als befahrbar anerkannt. Im Gegensatz zu unserem Szenario ist dieser Roboter in der Lage über verschiedene unbefestigte Untergründe zu fahren. Außerdem nutzt auch dieser Roboter mehrere Laser-Scanner für die Erkennung des Untergrundes.

In [13] wird die Navigation eines Roboters mittels Wegpunkten beschrieben. Als Sensorik stehen dafür die Odometrie des Roboters, ein Kompass, ein GPS-Empfänger und ein Gyroskop zur Verfügung. Es wird ein Verfahren vorgestellt, dass einen Kalman Filter verwendet um die Ungenauigkeiten des GPS-Empfängers auszugleichen. Da unser Roboter jedoch über kein Gyroskop verfügt und der Kompass des Smartphones störungsanfällig ist, stellt dieses Verfahren für uns keine Option dar.

Der mobile Außenroboter KURT2 [14] fährt in ähnlichen Umgebungen wie CampusGuide und verwendet ebenfalls Sensoren wie Kamera, Laser-Entfernungsmesser und einen GPS-Empfänger. Durch die Berechnung der GPS-Koordinaten, wird ein Weg von einem Startort zu einer Zielposition bestimmt. Das Ziel ist es hierbei, die Sensordaten in Echtzeit auszuwerten. Der Roboter wurde in dem Botanischen Garten der Universität Osnabrück getestet. Dabei fuhr er auf Wegen mit Pflastersteinen oder Schotterwegen, die durch Steine oder Rasen abgegrenzt sind. Angewendet wurden hier ein Randextraktionsalgorithmus und ein Gegenstandsextraktionsalgorithmus, um die Befahrbarkeit der Wege zu bestimmen. Der Weg wird zunächst als einzelnes Objekt aufgefasst und die Wegränder werden als Kanten herausgefiltert. Schwierigkeiten haben sich bei Veränderlichkeiten der Umgebung, unscharfen Fahrbahngrenzen, bei verschiedenen Beleuchtungen und sehr dynamischen Situationen, die schnelle Kontrollzyklen der Software erforderten, ergeben.

Für autonome und mobile Roboter ist es wichtig, befahrbare und nicht befahrbare Bereiche unterscheiden zu können. Dafür wurde eine Segmentierungsmethode entwickelt, die Farben und Texturen beschreibt und die befahrbaren Bereiche anzeigt [15]. Diese Methode nennt man Texturdeskriptor. Der Deskriptor arbeitet in mehreren Schritten. Als Erstes stellt er die Texturinformationen zusammen und führt den k-Means Algorithmus [16] aus, der ein Clustering aus den gegebenen Texturinformationen bildet. Anschließend wird ein Clustering Histogramm hergestellt, der letztendlich die Ergebnisse des Algorithmus präsentiert. Dieses Segmentierungsverfahren hat den Vorteil die Wiedererkennung der befahrbaren Bereiche zu ermöglichen. Der Roboter lernt verschiedene Typen von Wegen, durch Charakterisierung der Farben und Texturen, zu unterscheiden und auf ihnen zu navigieren.

In [17] wird ein Algorithmus angewendet, der anhand der Kameradaten

die Rauheit, Steigung, Unterbrechungen und Härte eines Geländes erkennt. Um die Oberfläche zu klassifizieren, wird zuerst eine Horizontlinie benötigt, die durch die Pixelsignatur der Grundebene bestimmt wird. Die Steigung eines Geländes wird ermittelt, indem auf den Bildern von zwei versetzt positionierten Kameras nach korrelierenden Punkten gesucht wird. Bei sehr homogenen Flächen treten allerdings Fehler in der Flächenerkennung auf, da die Steigung, ohne eindeutige Referenzpunkte nicht korrekt ermittelt werden kann. Geländeabbrüche, wie Treppen und Schlaglöcher, werden von der Flächenerkennung oft als Steigungen und nicht als Hindernisse interpretiert, was den Roboter in gefährliche Situationen bringen kann. Durch eine Kamera, die direkt vor dem Roboter den Boden aufnimmt, wird die Geländehärte bestimmt, was dem Roboter ermöglicht, Flächen mit zu lockerem Untergrund auszuweichen. Dieser Ansatz wird dadurch eingeschränkt, dass wechselnde Lichtverhältnisse und Schatten Fehler in der Erkennung des Geländes verursachen.

2.3 Mensch Maschine Interaktion

2.3.1 Personendetektion

Methoden der Personendetektion sind für Roboter insbesondere in Situationen notwendig, in denen sie Menschen finden und lokalisieren sollen. Viele Ansätze auf diesem Gebiet beschäftigen sich mit der Detektion von Menschen in Kamerabildern. Dies ist schwierig, da Menschen oft nicht eindeutig vom Hintergrund unterscheidbar sind. Außerdem ist das Erscheinungsbild von Menschen sehr variabel. Problematisch ist zudem, dass die Personendetektion sehr zeitaufwändig sein kann. Einige Ansätze, die sich mit diesem Thema befassen, werden im Folgenden vorgestellt.

Eine Möglichkeit, um Menschen zu finden, basiert auf der Tatsache, dass jeder Mensch bestimmte Körperteile hat, die jeweils identisch angeordnet sind [18]. Hier sollen verschiedene Körperteile in einem Bild identifiziert werden, die klar voneinander unterscheidbar sind. Zu diesem Zweck wird das Bild nach bestimmten Segmenten untersucht, die den gesuchten Körperteilen ähneln, und danach nach einer Kombination der gefundenen Körperteile gesucht, die zusammen einem anatomisch korrekt zusammengesetzten Menschen entsprechen.

Bei diesem Ansatz gibt es mehrere Probleme. Zum einen sich die Extremitäten in ihrer Form sehr ähnlich. Zum anderen funktioniert das Verfahren, so wie in [18] beschrieben, nur bei unbekleideten Personen, die in wenigen Standardpositionen stehen, mit zufriedenstellenden Ergebnissen. Ansonsten

leiden sowohl die Geschwindigkeit, als auch die Qualität der Ergebnisse. Dadurch ist dieses Verfahren ungeeignet für die Verwendung im Rahmen der Projektgruppe.

Ein Verfahren, das in Echtzeit Menschen detektieren und verfolgen soll, indem Veränderungen der Umgebung analysiert werden, wird in [19] untersucht. Damit dies auch unter schwierigen Bedingungen, wie etwa Umgebungen mit sich bewegenden Bäumen, sich verändernden Schatten oder nachts, funktioniert, werden mittels Hintergrundsubtraktion Unterschiede zwischen zwei aufeinander folgenden Bildern bestimmt und Menschen anhand von Informationen, bzgl. Größe und Farbe identifiziert. Dieser Ansatz weist auch in schwierigen Umgebungen eine hohe Klassifizierungsrate auf scheitert jedoch oftmals bei Personengruppen. Der Nutzen für die Projektgruppe ist trotzdem eher gering, da sich der Roboter häufig bewegt. Es ist jedoch für diesen Ansatz wichtig, Regionen über einen längeren Zeitraum zu beobachten, um den Hintergrund zu erfassen. Da unsere Anwendung jedoch echtzeitfähig sein soll, wurde dieser Ansatz nicht weiter verfolgt.

Ein sehr robuster Ansatz zur Detektion von Gesichtern wird in [20] betrachtet. Dieser verwendet Integralbilder, um möglichst effizient die folgenden Bearbeitungen durchzuführen. Integralbilder zeichnen sich dadurch aus, dass in einem Pixel die Summe der Farbwerte aller Pixel steht, die sich im Rechteck zwischen dem Pixel und dem Ursprung befinden. Somit kann zur Laufzeit die Summe der Pixel innerhalb eines Rechtecks nur mit Hilfe der Eckpunkte dieses Rechtecks berechnet werden. Weitere Eigenschaften, die den Ansatz auszeichnen, sind die Verwendung von einfachen Rechteckmerkmalen, sowie einer Kaskade. Durch die Abweisungskaskade wird sicher gestellt, dass uninteressante Bildregionen schnell von der Klassifikation ausgeschlossen werden. Dafür werden zuerst schwache Klassifikatoren angewandt, die wenig Rechenaufwand erfordern. Starke und rechenaufwändige Klassifikatoren kommen erst spät zum Einsatz und werden daher nur auf wenige Bildregionen angewandt. Dieses Verfahren hat sich als sehr schnell herausgestellt und ermöglicht es in Echtzeit Gesichter in Bildern effektiv zu detektieren. Deshalb halten wir diesen Ansatz für die Zwecke der PG geeignet.

In [21] wird ein Verfahren beschrieben, das auf [20] aufbaut. Hierbei werden Informationen bezüglich des Erscheinungsbildes und der Bewegung benutzt, um Menschen in Videosequenzen zu detektieren. Dies funktioniert auch unter schwierigen Bedingungen wie Regen. Dieser Ansatz ist sehr effizient und weist nur eine geringe Rate an falsch positiven Detektionen auf. Die Schwäche des Verfahrens besteht darin, dass nicht vollständig sichtbare Menschen schlechter gefunden werden als komplett sichtbare Menschen. In unserem Szenario befinden sich die zu detektierenden Personen oft nah am Roboter, wodurch sie nicht komplett von der Kamera erfasst werden können.

Aus diesem Grund ist dieses Verfahren nicht für unseren Anwendungsfall geeignet.

Unter der Verwendung von Histogrammen von orientierten Gradienten (HoG) können Menschen schnell und zuverlässig erkannt werden [22]. Ein HoG zählt die Vorkommnisse bestimmter Orientierungen von Gradienten in ausgewählten Bildausschnitten und beschreibt so die Form der gesuchten Objekte (hier Menschen), um diese zu erkennen. Dabei werden Blöcke in unterschiedlichen Größen und Größenverhältnissen betrachtet, um mehr Möglichkeiten zur Detektion zu erhalten. Zusätzlich wird eine Abweisungskaskade verwendet, bis sowohl eine vorher festgelegte minimale Detektionsrate, als auch eine maximale falsch positive Rate erfüllt ist. Da uninteressante Bildausschnitte sehr schnell verworfen werden, ist die Personendetektion mit diesem Verfahren in Echtzeit möglich, wobei die Detektionsrate sehr hoch war. Dieser Ansatz erscheint uns ebenfalls für die Anforderungen der PG an die Personendetektion geeignet.

Ein Verfahren, das den HoG-Ansatz verbessert, wird in [23] vorgestellt. Dabei werden zusätzlich Informationen bzgl. Kleidung, Texturen und Farbe in die Auswertung einbezogen. Da durch diese zusätzlichen Informationen die Anzahl der Dimensionen derart steigt, dass diese nicht mehr effizient verarbeitet werden können, werden unvollständige kleinste Rechtecke verwendet, um die Anzahl der Dimensionen zu reduzieren. Dies geschieht durch eine Projektion eines Merkmalsvektors, der aus einem Detektionsfenster extrahiert wurde, auf einen Gewichtsvektor, wobei das Ergebnis ein Eigenvektor ist, welcher zur Klassifikation verwendet wird. Der Ansatz besticht durch sehr hohe richtige Detektionsraten bei sehr hoher Effizienz und ist, verglichen mit anderen Verfahren, jedem gängigen Ansatz überlegen. Die Autoren stellen auf ihrer Internetseite[1] ein Programm zur Verfügung, das dieses Verfahren verwendet. Dieses konnte unsere Erwartungen jedoch nicht erfüllen, da die Gesichtsdetektion mit diesem Programm selbst auf niedrig aufgelösten Bildern nicht in Echtzeit lief, weshalb dieser Ansatz nicht weiter verfolgt wurde.

2.3.2 Emotionen bei Robotern

Die Interaktion von Robotern und Menschen ist ein noch junges Forschungsgebiet, welches erst seit den 90er Jahren des letzten Jahrtausends besteht. Hierzu gibt Dautenhahn einen guten Überblick über das Forschungsgebiet [24]. Sie beschreibt die Entwicklung des gesamten Forschungsgebietes und verschiedene Forschungsprojekte, welche sich mit Emotionsausdrücken in der Interaktion von Robotern und Nutzern beschäftigen. Ebenso ist eine Klas-

[1] http://www.umiacs.umd.edu/∼schwartz/softwares.html#detectorPLS

sifikation von Robotern gemäß den von ihnen benötigten Fähigkeiten zum Ausdruck von Emotionen enthalten. Diese Klassifikation reicht von Robotern, die mangels direktem Kontakt mit Menschen keinerlei Fähigkeiten zum Emotionsausdruck verfügen müssen (beispielsweise Weltraumsonden), über Tourguides, deren Arbeitsgebiet nach einer mittelmäßige Fähigkeit zur Emotionsexpression verlangt, bis hin zu „Robotergehilfen“ (*Robot Companions*), die mit Menschen zusammen leben. Diese dienen z.B. als Helfer für behinderte Personen. Ein solches Einsatzgebiet erfordert, dass die Fähigkeit des Roboters zum Emotionsausdruck entsprechend hoch sein muss, um eine größtmögliche Akzeptanz zu gewährleisten.

Speziell auf dem Gebiet der Tourguides (welche CampusGuide von ihrem Aufgabengebiet her am Nächsten kommen) haben sich die emotionalen Kapazitäten der Roboter in den letzten 10 Jahren stark verbessert. Während anfänglich Tourguides benutzt wurden, welche nicht über die Möglichkeit zur Expression von Emotionen verfügten, stellte man schnell fest, dass ein emotionales Verhalten bei Robotern es diesen erleichtert, bei den Benutzern ein erwünschtes Verhalten hervorzurufen.

Ein gutes Beispiel für die Auswirkung von Emotionen bei Robotern ist im Vergleich zwischen den Tourguides Rhino und Minerva zu sehen. Bei Rhino handelte es sich um einen Tourguide, welcher im deutschen Museum in Bonn eingesetzt wurde, welcher über keine Fähigkeiten verfügte, Emotionen auszudrücken (siehe [7] und [1]).

Im Gegensatz dazu war Minerva, der Tourguide des Smithsonian Museums in Washington, mit internen Emotionszuständen ausgestattet (mehrere Stufen von Freude und Ärger) und verfügte über ein „Gesicht“ mit ausmodelliertem mechanischen Mund sowie Augenbrauen, um diese Emotionen der Umwelt mitzuteilen. Emotionen wurden von Minerva vor allem dann eingesetzt, wenn Menschen ihm den Weg versperrten und er diese zum beiseite treten motivieren wollte.

Ein Vergleich der beiden Tourguides ergab, dass die Durchschnittsgeschwindigkeit von Minerva über jener von Rhino lag, obwohl Minerva einen Arbeitsplatz hatte, an dem viel mehr Publikumsverkehr war, als am Arbeitsplatz von Rhino. Auch gab es bei Minerva eine erheblich niedrigere Anzahl an Fällen, in denen Besucher bewusst versuchten, destruktives Verhalten gegenüber dem Roboter auszuüben, z.B. durch den Versuch, den Roboter in unbekannte Bereiche, wie etwa das Museums-Treppenhaus, zu drängen, oder ihm bewusst den Weg durch Umrunden zu versperren und damit seine Pfadplanung zum Absturz zu bringen (siehe [25]).

Kapitel 3

Grundlagen

In der Robotik werden die Zielsetzungen und die Möglichkeiten eines Projektes sehr stark von den Eigenschaften des benutzten Systems beeinflusst. Dies umfasst sowohl alle Merkmale der verwendeten Hardware als auch der Software, die auf dieser zum Einsatz kommt. Daraus ergeben sich einige Einschränkungen und Besonderheiten, die bei der Realisierung des Projektes beachtet werden müssen. In diesem Kapitel wird der verwendete Roboter mit allen Eigenschaften und Möglichkeiten erläutert. Dazu gehören die grundlegende Roboterplattform, die eingebauten Sensoren und das angeschlossene Smartphone. Außerdem beschreibt dieses Kapitel die Eigenschaften des Player/Stage Software-Frameworks, welches die Steuerungsbefehle auf dem Roboter ausführt und einige grundlegende Funktionen bereitstellt.

3.1 Hardware

Die Hardware des verwendeten Roboters besteht aus drei wesentlichen Teilen, die unterschiedliche Aufgabenbereiche erfüllen. Dabei realisiert die Roboterplattform SCITOS G5 die Bewegung des Roboters und bietet die Ressourcen für die nötigen Berechnungen. Durch das angeschlossene Smartphone HTC Legend wird ermöglicht die Position mit GPS zu bestimmen und SMS zu senden und zu empfangen. Um die Umgebung um den Roboter zu erkunden, wurden mehrere Sensoren mit verschiedenen Eigenschaften angeschlossen. Die Abbildung 3.1 zeigt den Roboter mit den Sensoren. In den folgenden Abschnitten betrachten wir die einzelnen Hardwarekomponenten im Detail.

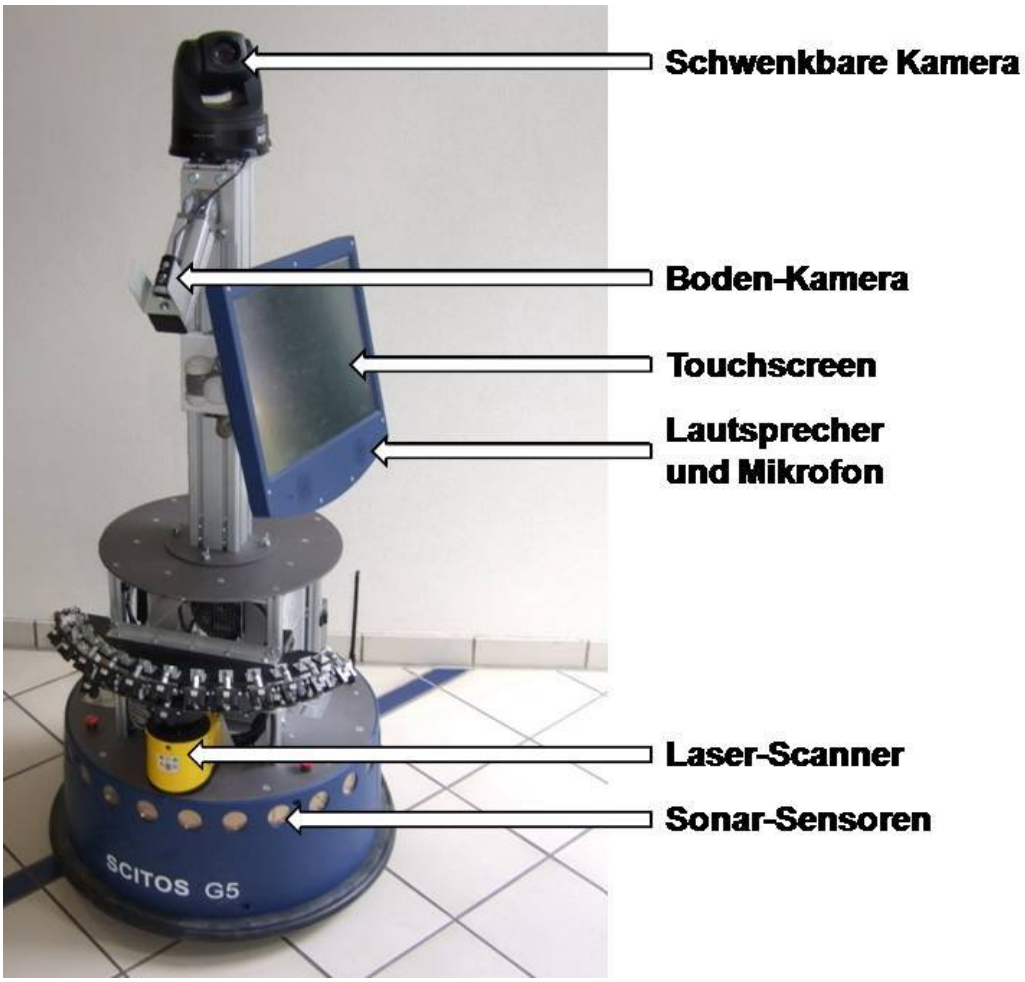


Abbildung 3.1: SCITOS G5

3.1.1 Roboterplattform

Als Grundgerüst des Roboters dient uns die Roboterplattform SCITOS G5 der Firma MetraLabs GmbH [1]. Sie wurde als eine flexible Hardware-Architektur für mobile Roboter entwickelt, die insbesondere bei Forschungsprojekten verwendet wird. [26]. Mobile Roboter haben zwar verschiedene Anforderungen, die durch die unterschiedlichen Ziele und Aufgaben der Roboter entstehen, aber viele Bestandteile sind meistens die Gleichen. Dadurch empfiehlt sich die Verwendung einer flexiblen Plattform, die die gemeinsamen Komponenten beinhaltet. So müssen diese Teile nicht für jeden Roboter neu entwickelt werden. Zu diesen Komponenten gehören zum Beispiel die Stromversorgung und die Kommunikation zwischen den Bestandteilen des Roboters. Zusätzliche Komponenten, wie z. B. Sensoren und Interaktionsmöglichkeiten mit dem Benutzer, vervollständigen den Roboter und passen ihn an die Anforderungen der Projektgruppe an.

Die Plattform SCITOS G5 ist nur für den Einsatz in geschlossenen Räumen konzipiert. Dadurch kann der Roboter nur auf flachen und festen Oberflächen fahren. Auf unebenen Flächen, sowie auf losem Untergrund kann er stecken bleiben. Da der Einsatzbereich des Roboters bei den Aufgaben der Projektgruppe außerhalb von geschlossenen Räumen liegt, kann er auf Flächen stoßen, die er nicht befahren kann. Auf dem Campus sind an vielen Stellen Flächen mit Gras oder loser Erde direkt neben den Wegen. Um dort nicht stecken zu bleiben muss vermieden werden, dass der Roboter auf solche Flächen fährt. Außerdem hat der Roboter keine Möglichkeit Stufen oder große Steigungen zu überwinden. Diese Hindernisse könnten zu einem Umstürzen des Roboters führen und müssen auf jeden Fall vermieden werden. Durch diese Einschränkungen ist die Roboterplattform nur sehr bedingt für den Einsatz im Freien geeignet. Da der Einsatzbereich des Roboters in dieser Projektgruppe allerdings im Freien liegt, müssen Wege gefunden werden, um den Roboter vor diesen Hindernissen zu schützen.

Um die anfallenden Berechnungen auf dem Roboter durchzuführen, besitzt er einen eingebauten PC. Dieser regelt außerdem die Steuerung des Roboters und speichert alle aufgenommenen Sensordaten. Dieser PC ist mit einem Intel Core Duo T2300 1,6 GHz ausgestattet. Er besitzt 2048 MB RAM und eine 100-GB-Festplatte. Um die Echtzeit-Fähigkeit des Systems nicht zu gefährden, können keine Berechnungen ausgelagert werden. Daher müssen alle Funktionen des Roboters, sowie zusätzliche Berechnungen unseres Systems, auf diesem PC laufen. Dabei muss zusätzlich beachtet werden, dass dadurch die Rechenleistung des PCs nicht überschritten wird, da auch dies einen Echtzeit-Einsatz gefährden würde. Zusätzlich ist der PC mit einer

[1] http://www.metralabs.com/

WLAN-Antenne ausgerüstet, die es dem Roboter ermöglicht jederzeit Daten zu versenden und zu empfangen.

3.1.2 Sensoren

Der Roboter ist mit verschiedenen Sensoren ausgerüstet, mit denen er die Umgebung erfassen kann. Dabei handelt es sich um ein Odometriesystem, Sonar-Sensoren sowie einen Laser-Entfernungsmesser. Weiterhin ist die Plattform mit einem *Bumperring* und zwei Kameras ausgestattet. Im folgenden Abschnitt werden die Eigenschaften der Sensoren beschrieben.

Die Odometrie des Roboters erfasst die Bewegungen der Räder des Roboters und bildet die eigene Position und Richtung relativ zu einem Startpunkt in einem internen Koordinatensystem ab. Abweichungen bei der Messung der Radbewegungen (z. B. durch Verlust der Bodenhaftung eines der Räder, unterschiedliche Abnutzung der Räder, ...) können zu einer ungenauen Abbildung der Roboterbewegung führen. Deshalb sind Maßnahmen notwendig, die diese Fehler korrigieren.

Auf der Roboterplattform sind insgesamt 24 Sonar-Sensoren verbaut, die den Nahbereich um den Roboter abtasten. Zusammen decken die Sonar-Sensoren einen Winkel von 360° um den Roboter ab und haben eine maximale Reichweite von 3 m. Die Messgenauigkeit nimmt mit ansteigender Entfernung stark ab, weshalb nur Hindernisse nah um den Roboter erkannt werden können.

Eine zuverlässigere Abtastung der Umgebung liefert der Laser-Entfernungsmesser. Der Abtastwinkel von 270° wird allerdings durch die Installation auf dem Roboter auf einen Bereich von 180° in Fahrtrichtung des Roboters eingeschränkt. Mit einer Abtastauflösung von 0,5° und einer Reichweite von 30 m liefert der Laser-Scanner ein sehr genaues Abbild der Umgebung. Durch die hohe Reichweite kann der Laser-Scanner Hindernisse schon früh erkennen, wodurch genügend Vorlauf zur Berechnung einer Ausweichbewegung bereit steht. Durch seine waagerechte Montage in einer Einbauhöhe von 40 cm über dem Boden, kann der Laser-Entfernungsmesser nur Hindernisse in einer Ebene parallel zum Boden erkennen. Auf Grund dieser Einschränkung können keine Hindernisse, die kleiner als 40 cm sind oder die höher als 40 cm beginnen erkannt werden.

Der SCITOS G5 besitzt einen *Bumperring* um den gesamten Rumpf des Roboters. Stößt der Roboter mit dem Bumper gegen ein Hindernis wird der Antrieb sofort abgeschaltet. So ist sicher gestellt, dass der Roboter nicht beschädigt wird oder keine Personen und Gegenstände in der Umgebung beschädigt, wenn er einmal einem Hindernis nicht rechtszeitig ausweicht.

Ungefähr auf Augenhöhe befindet sich eine dreh- und neigbare Kamera

mit PAL-Auflösung (720x576 Bildpunkte). Sie hat einen Dreh-Bereich von -170° bis +170° und einen Neig-Bereich von -30° bis +90°, wodurch die Kamera aus fast jeder Richtung Bilder aufzeichnen kann, ohne dass die Plattform bewegt werden muss.

Zusätzlich ist auf dem Roboter noch eine zweite Kamera mit einer Auflösung von 640x480 Bildpunkten installiert. Sie ist in einem fest eingestellten Winkel von 45° nach unten in Fahrtrichtung vor den Roboter gerichtet. Diese Kamera ermöglicht es, den Untergrund und kleine Hindernisse vor dem Roboter zu erkennen.

Für die Interaktion mit dem Benutzer ist auf dem Roboter ein Touchscreen mit eingebauten Lautsprechern und einem Mikrofon installiert. Auf dem Touchscreen können dem Benutzer verschiedene Informationen angezeigt und Eingaben entgegen genommen werden.

3.1.3 Smartphone

Um zusätzliche Interaktionsmöglichkeiten über die Benutzung des Touchscreens hinaus zu bieten, haben wir ein Smartphone an den SCITOS G5 angeschlossen. Über dieses kann der Benutzer seine aktuelle Position an den Roboter schicken, wodurch CampusGuide die Position des Nutzers bestimmen und zu ihm fahren kann. Als Smartphone nutzen wir das HTC Legend (siehe Abb. 3.2). Es bietet mehrere Funktionen, die wir für die Projektgruppe nutzen. Die wichtigste Funktion, die uns das Handy liefert, ist die Kurznachrichten Funktion (*Short Message Service*, SMS), welche die Kommunikation mit dem Benutzer über große Distanz ermöglicht. Zusätzlich bietet uns das Smartphone die Nutzung eines globalen Navigationssystems (*Global Positioning System*, GPS). Mit den GPS-Koordinaten, die CampusGuide von dem Smartphone erhält, kann sich der Roboter auf dem Campus lokalisieren und von der aktuellen Position aus eine Route zum Ziel ermitteln. Außerdem bietet das HTC Legend eine *Bluetooth*-Schnittstelle. Dies stellt eine Datenverbindung zwischen dem Roboter und dem Handy bereit und bietet die Möglichkeit, die Entfernung zu anderen *Bluetooth*-Geräten ermitteln.

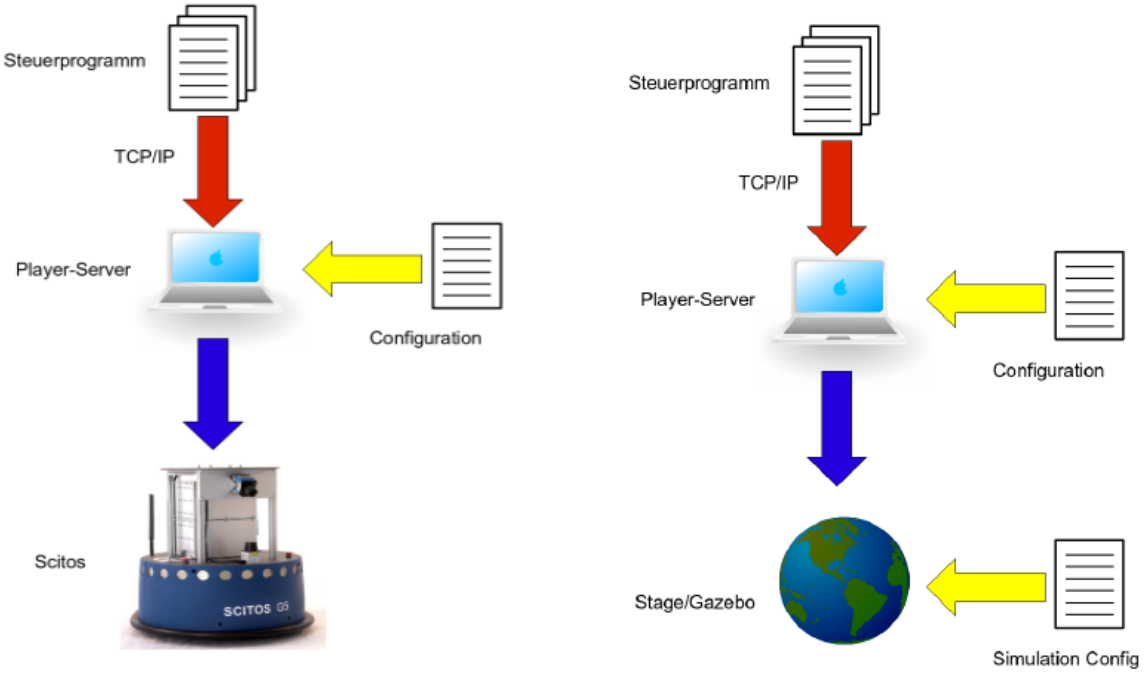
Abbildung 3.2: Smartphone - HTC legend

3.2 Player/Stage

In diesem Abschnitt möchten wird das Framework Player/Stage vorgestellt. Es ist ein plattform- und sprachunabhängiges Framework für die Steuerung von mobilen Robotern, welches keine Annahme über den benutzten Roboter macht. Das Framework bietet einen Roboterserver. Zu Simulationszwecken wird eine Simulationsumgebung mitgeliefert. Im Weiteren betrachten wir die beiden Werkzeuge getrennt voneinander.

Player bietet eine Schnittstelle zu den Sensoren eines Roboters. Dabei erlaubt das umgesetzte Client/Server Modell dem Client eine Verbindung über TCP-Sockets zum Player aufzubauen, die Daten von den Sensoren zu lesen, Anweisungen zu geben und die Sensoren online zu konfigurieren. Der Player kommuniziert mit den Sensoren über dazugehörige Treiber. Dem Client dagegen werden Schnittstellen für die Kommunikation mit den Sensoren zur Verfügung gestellt. Anschaulich interagiert der Player beispielsweise mit einem Lasertreiber, stellt dem Client jedoch nur eine Laserschnittstelle zu Verfügung. Die Struktur erlaubt, den Client auf mehreren Robotern laufen zu lassen. Auf der anderen Seite erlaubt der Player-Server, gleichzeitig mehrere Clientverbindungen aufzubauen. Die vereinfachte Architektur vom Player-Server und dem Simulationstool ist in der Abbildung 3.3 dargestellt.

Eine große Anzahl von abstrakten Treibern sind in dem Player-Server verfügbar und werden ständig von der Benutzergemeinschaft weiter entwi-



(a) Player-Server (b) Stage

Abbildung 3.3: Player/Stage

ckelt. Ein abstrakter Treiber kommuniziert nicht direkt mit der Hardware, wie es die Treiber in der Player-Umgebung sonst tun, sondern benutzt andere Treiber. Die wichtigste Verwendung der abstrakten Treiber ist die Umsetzung von Algorithmen, wie es beispielsweise in der Navigation getan wird. Dort werden Treiber benutzt die Hindernisvermeidung, oder auch Umfahrung ermöglichen. Es existieren bereits mehrere abstrakte Treiber.

Der Player-Server wird mit der Ausführung des Konfigurationsfiles gestartet. In dem Konfigurationsfile wird definiert welche Treiber instantiiert werden sollen und wie sie an die Hardware gebunden werden. Daraufhin wird das Programm des Users in dem Client ausgeführt und verbindet sich mit dem Roboterserver zur weiteren Interaktionen.

Das Stage Framework bietet einen Robotersimulator an. Dieser Simulator ist in der Lage mehrere Roboter gleichzeitig in einer zweidimensionalen Umgebung zu simulieren.

Stage interagiert als eine Schnittstelle von Player, weshalb meistens keine Änderung notwendig sind, um nach der Simulation die wirkliche Anwendung auszuführen. Der Simulator bietet keine Unterstützung der Treiber wie den

oben erwähnten Lasertreiber an, sondern er hat ein abstraktes Treibermodell wie zum Beispiel *laser*. In Gegensatz zur Konfigurationsfile von Player gibt es in Stage eine sogenannte World-Datei. Hier wird das abstrakte Modell definiert. Letztendlich nimmt Stage die Anweisungen von Player an und simuliert sie in der erschaffenen Modellwelt.

Kapitel 4

Realisierung

4.1 Die Softwarearchitektur

Gemäß den Anforderungen an die Projektgruppe handelt es sich bei Campus-Guide um ein verteiltes System. Es besteht zum einen aus dem SCITOS G5, der Sensordaten verarbeiten, Wege planen sowie mit dem Benutzer durch den Touchscreen interagieren muss. Zusätzlich agieren die Algorithmen und Programme aus Player/Stage als eigenständige Module. Als zweite Komponente existiert das Smartphone, das die SMS-Kommunikation mit dem Benutzer, die Bluetooth-Abstandsmessung sowie die Webanbindung des Roboters realisiert. Als dritte Komponente kommt der Webserver hinzu, der von Benutzern wie Administratoren verwendet wird, um auf Daten des Roboters zuzugreifen.

Die Kommunikation der einzelnen Komponenten des Systems, insbesondere mit den fertigen Modulen von Player, stellt eine große Schwierigkeit dar. Eine Lösung für dieses Problem bildet das Blackboard von Player/Stage. Es bietet den Vorteil nativ mit Player-Modulen kommunizieren zu können und stellt zusätzlich eine zentrale, einfach zu erreichende Datenstruktur dar, die Vorteile für ein verteiltes System bietet. Da sich auf dem Blackboard beliebige Datentypen ablegen lassen, auf die von allen anderen Programmteilen aus zugegriffen werden kann, stellt es eine ideale zentrale Komponente dar. Wir haben uns daher entschlossen die Kommunikation aller Module über das Blackboard zu realisieren.

Der Zugriff von Modulen auf Daten im Blackboard wird mit einer *Observer*-Architektur (siehe [27]) realisiert. Ein Modul, das auf einen bestimmten Wert zugreift, wird als *Observer* für diesen Wert eingetragen und über Änderungen im entsprechenden Datenfeld auf dem Blackboard informiert.

So wird z.B. die, vom Smartphone bestimmte, aktuelle GPS-Position des

Roboters auf das Blackboard geschrieben. Alle Module, die mit diesem Wert arbeiten, also z.B. der Webserver und die Navigation, werden über diese Änderung informiert und können entsprechend reagieren. Welche Datenfelder auf dem Blackboard existieren und welches Modul auf welches Feld zugreift, ist in Abbildung 4.1 dargestellt.

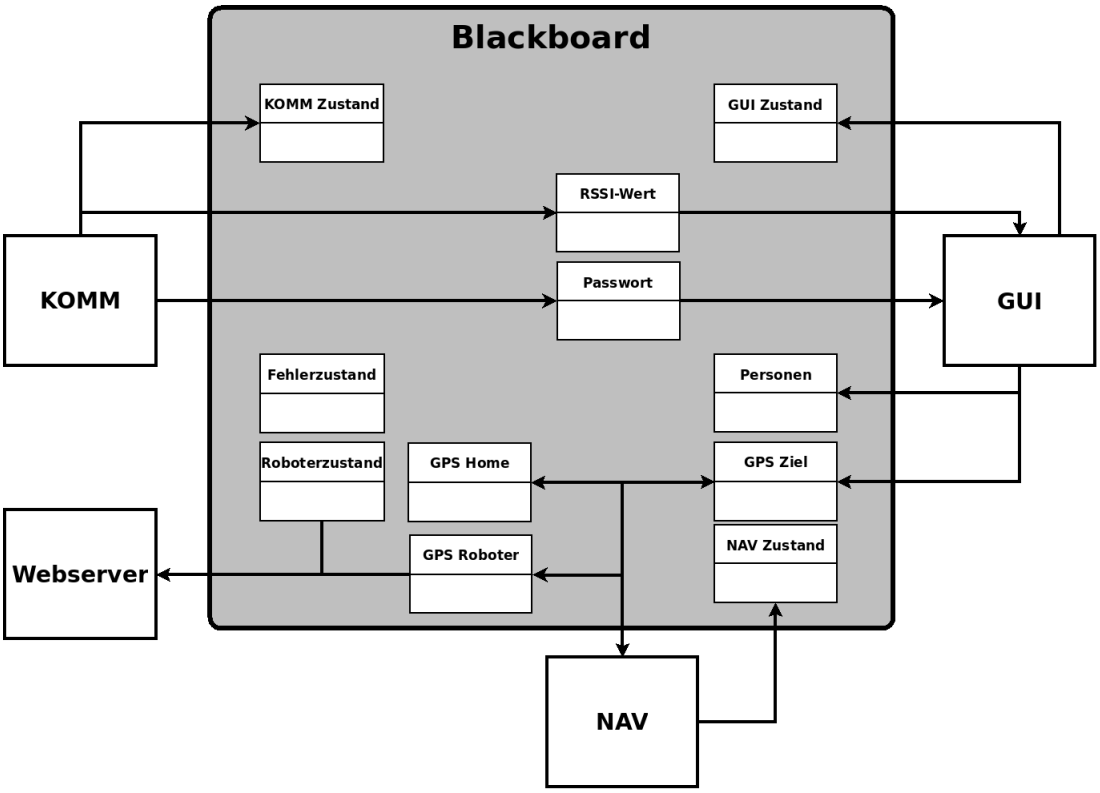


Abbildung 4.1: Die Datenfelder im Blackboard und der Zugriff der einzelnen Module auf diese. Manche Module greifen nur lesend auf Datenfelder zu, andere hingegen schreibend. Zum Beispiel schreibt das KOMM-Modul das Passwort auf das Blackboard, während dieses vom GUI-Modul nur gelesen wird. Die Art des Zugriffs wird durch die Richtung der Pfeile repräsentiert. Der Fehlerzustand und der Roboterzustand werden implizit von allen Modulen gelesen.

Zusätzlich werden auf dem Blackboard Zustands- und Fehlerwerte verwaltet. Diese geben an, in welchen Zuständen sich das System und die einzelnen Module momentan befinden und ob ein Fehler in einem der Module vorliegt. Der Wert *Roboterzustand* (siehe Tabelle 4.1) gibt z.B. an, ob der Roboter sich gerade auf dem Weg zum Benutzer befindet, momentan ohne Aufgabe ist oder ob ein Fehler vorliegt. Eine komplette Liste aller Zustandscodes ist

Roboterzustand		
Wert	Zustand	Beschreibung
0	Idle	Der Roboter ist einsatzbereit
1	Busy	
2	Moving to Client	Der Roboter fährt zu einem Benutzer
3	At Client Pos	Der Roboter hat die vom Benutzer angegebene Position erreicht
4	Moving to Target	Der Roboter fährt ein vom Benutzer gewähltes Ziel an
5	At Target	Das vom Nutzer angegebene Ziel wurde erreicht
6	Moving Home	Nach beendeter Navigation fährt der Roboter seine Ladestation an
7	Maintenance Mode	Wartungsmodus. Der Roboter nimmt keine Anfragen von Benutzern an
8	Error Mode	Ein Fehler ist aufgetreten, die genaue Art des Fehlers ist im Datenfeld „Fehlercode“ gespeichert
9	Moving to exact user pos	Benutzer wird mit Personenerkennung gesucht
10	At exact user pos	Benutzer wurde identifiziert und interagiert

Tabelle 4.1: Die Werte die das Datenfeld *Roboterzustand* im Blackboard annehmen kann mit ihrer Bedeutung.

in Anhang A zu finden.

Unser System besteht im Wesentlichen aus vier Modulen, deren Realisierung in diesem Kapitel erläutert wird: das Kommunikationsmodul (KOMM), das die Kommunikation zwischen Roboter und Benutzer über das Smartphone realisiert; das Modul Mensch-Maschine-Interaktion (MMI), welches das Interface für den Benutzer, sowie das Erkennen von Personen realisiert; das Navigationsmodul (NAV), das sich mit der Lokalisation des Roboters, Pfadplanung und Hindernisvermeidung beschäftigt und dem Webserver, der es ermöglicht die Postion des Roboters aus dem Internet zu verfolgen.

4.2 Kommunikationsarchitektur zwischen Nutzer, Smartphone und Roboter

Möchte ein Nutzer unseren mobilen Roboter in Gebrauch nehmen, müssen Kommunikationswege zwischen beiden Parteien existieren. Der erste Schritt besteht darin, dass der Nutzer CampusGuide kontaktiert. Hierzu benötigen wir eine Komponente, die als Mediator zwischen dem Nutzer und dem Roboter fungiert und somit die Kommunikation zwischen beiden realisiert. Als Kommunikationskomponente haben wir das im vorigen Kapitel aufgeführte Smartphone gewählt. Die Gründe hierfür sind zum einen, dass wir mit einem SMS-Service eine sehr einfache und für den Nutzer angenehme Form anbieten können unser System zu kontaktieren. Darüber hinaus bietet das gewählte Smartphone ein quelloffenes Betriebssystem, was es uns ermöglicht eigene Programme zu entwickeln und nach unseren Vorstellungen auf dem Telefon auszuführen. In Abbildung 4.2 sind die Kommunikationswege, vereinfacht dargestellt, zu sehen.

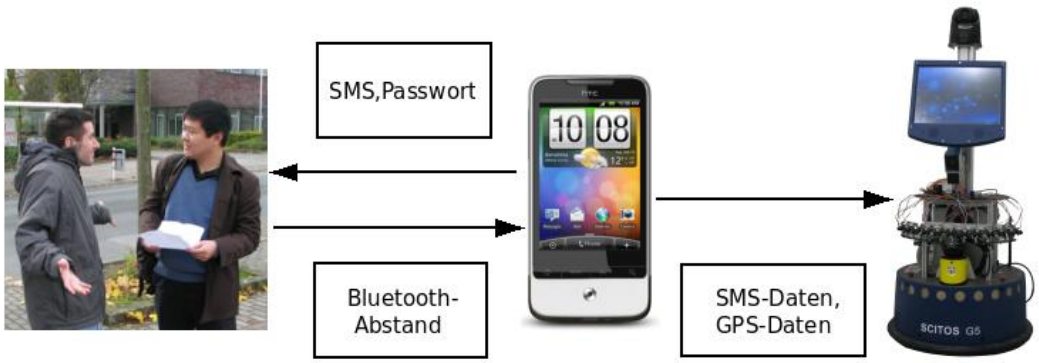


Abbildung 4.2: Übersicht der Kommunikation zwischen dem Benutzer, Smartphone und Roboter

4.2.1 Einführung in das Android-Betriebssystem

In diesem Kapitel werden die grundlegenden Methoden und Eigenschaften des Android-Betriebssystems[1] erläutert. Android ist ein Open Source Betriebssystem, das 2008 von Google speziell für mobile Geräte entwickelt wurde. Das System zeichnet sich durch einige wichtige Eigenschaften aus. So wird z.B. jedes laufende Programm in einer eigenen virtuellen Maschine,

[1] www.android.com

der sogenannten *Dalvik virtual machine* ausgeführt und besitzt einen nur diesem Prozess zugesicherten Speicherbereich. Zudem hat jedes Programm eigene Rechte. Der Kern des Systems basiert auf Linux und daher ist auch die Rechtevergabe ganz ähnlich der von Linux auf *permissions* aufgebaut. Eine weitere ganz wesentliche Eigenschaft des Android-Betriebssystems ist das Prinzip der lose gekoppelten Komponenten. Darunter ist zu verstehen, dass ganze Programme oder auch nur einzelne Teile davon in anderen Programmen verwendet werden können. So ist es beispielsweise denkbar, dass zwei völlig verschiedene Programme eine Komponente, welche z.B. Kontaktdaten bereitstellt, nutzen. Ganz generell sind Android-Programme aus vier verschiedenen Komponententypen aufgebaut. Diese sind *Activities*, welche die Benutzeroberfläche bereitstellen, *Services* welche Hintergrundprozesse bereitstellen, *Content provider* die die Daten der Anwendung verwalten, sowie *Broadcastreceiver*, die dazu da sind als Vermittler im gesamten System zu fungieren und auf bestimmte, vorher festgelegte, Nachrichten zu reagieren.

4.2.1.1 Intents

Das im vorigen Abschnitt angesprochene Prinzip der lose gekoppelten Komponenten wird durch die sogenannten *Intents* ermöglicht. Ein *Intent* ist eine Nachricht, deren Typ und Inhalt dynamisch festgelegt werden kann. Das heißt, man kann als Entwickler seine eigenen *Intents* erstellen, beliebige Daten anhängen und diese dann ins System schicken. Alle Komponenten die den Typ des *Intent* kennen und auf diesen Typ reagieren, bekommen die Nachricht und können darauf ihrerseits eine Aktion durchführen. Dadurch ist eine sehr einfache und effiziente Art der Interprozesskommunikation geschaffen, mit Hilfe derer man komplexe Systeme handhaben kann.

4.2.2 Bluetooth

Die Bluetooth-Verbindung wird im Android-Betriebssystem durch die Implementierung des BlueZ *Bluetooth-Stacks* repräsentiert. Dieser *Stack* bietet die grundlegenden Methoden um eine Verbindung zu erstellen. Um ein anderes Bluetooth-Gerät zu finden und eine Kommunikation zu initiieren müssen bestimmte Voraussetzungen erfüllt sein. Zum einem muss auf Seite des Servers ein Bluetooth-Dienst angeboten werden. Der Dienst muss hierfür dem lokalen *Service Discovery Protocol* (SDP) Server mit einem *Universally Unique Identifier* (UUID) übergeben werden. Zum anderen muss auf Seite des Clients genau dieser UUID bekannt sein, damit der richtige Dienst angefragt werden kann. Auf die Anfrage antwortet der Server mit den Spezifikationen der Verbindung und die Kommunikation kann beginnen. Typischerweise findet diese

Kommunikation über einen *Radio Frequency Communication* (RFCOMM) Kanal statt, welcher eine serielle Verbindung zwischen Bluetooth-Geräten emuliert.

4.2.3 Nutzerfindung per SMS

Wie im vorigen Kapitel schon angesprochen ist die primäre Kommunikation zwischen dem Nutzer und CampusGuide durch SMS realisiert. Weitere Möglichkeiten, die Kommunikation zum Nutzer zu realisieren, wären eine Ortung per Anruf oder über das Internet gewesen. Dagegen sprechen aber einige wichtige Gründe. Zum einen hat nicht jeder ein modernes Smartphone, um damit ins Internet zu gelangen und auch die Ortung per Anruf würde sich als sehr schwierig und vor allem unzureichend erweisen. Die Ortung per Anruf würde sich an den Telefonmasten orientieren um dadurch die ungefähre Position des anrufenden zu ermitteln. Dieses Verfahren ist allerdings sehr ungenau und hängt zu stark von den Telefonmasten ab, so dass der Roboter den anrufenden nur in den seltensten Fällen erreichen würde. Der Hauptgrund für die Wahl von SMS ist, dass wir über eine SMS sehr einfach Informationen, in Form des Standortes und des Telefonnamens, vom Benutzer erhalten und vor allem, dass wir sehr einfach notwendige Daten, in Form eines Passwortes, an den Benutzer zurücksenden können. Die Realisierung des SMS-Kommunikationsprogramms wird im Folgenden näher erläutert.

Wenn eine vom Nutzer gesendete SMS ankommt, wird sie von einem Programm auf dem Smartphone registriert und bearbeitet. Abbildung 4.3 veranschaulicht die Verarbeitung einer SMS vom Nutzer. Das Verschicken der Daten am Ende der Bearbeitung gibt anderen Komponenten die Möglichkeit diese zu nutzen, um so z.B. den Standort in einer Datenbank nachzuschlagen.

4.2.4 GPS

Der von uns verwendete Roboter besitzt kein eigenes GPS-Modul. Aus diesem Grund benutzt CampusGuide das Smartphone zur Lokalisierung. Die Daten müssen demnach erst vom Telefon auf den Roboter geschickt werden, allerdings haben wir durch das Smartphone den Vorteil, dass zusätzlich ein digitaler Kompass zur Verfügung steht. Mithilfe des Kompasses ist es möglich die aktuelle Orientierung des Roboters festzustellen, um so die Qualität der Navigation und die Präzision der Lokalisierung zu verbessern. Die Funktionalitäten werden in einem Modul für das Smartphone realisiert, welches ständig die aktuelle Position des Roboters abruft und diese dann mit

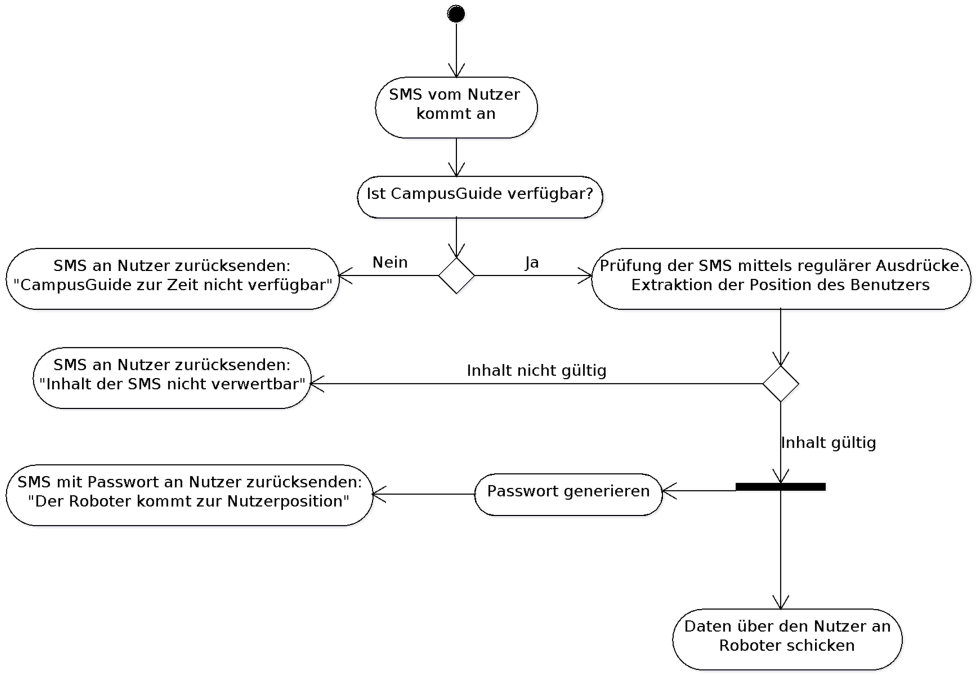


Abbildung 4.3: Ablauf der Verarbeitung einer SMS vom Benutzer

der entsprechenden Orientierung an das Blackboard sendet. Hinzu kommt die aktuelle Systemzeit, damit die Positionen von CampusGuide zu jeder Zeit voneinander unterschieden werden können, selbst wenn der Roboter sich nicht bewegt.

4.2.5 Bluetooth-Abstandsmessung

Hat ein Benutzer des CampusGuide eine Führung gestartet, so gilt es, jenen auf dem Weg zum gewählten Ziel nicht zu verlieren. Hierzu muss der mobile Roboter den Abstand zu der geführten Person messen, ohne dass dem Nutzer ein zusätzliches Gerät übergeben wird. Da moderne Mobiltelefone meist über ein Bluetooth-Modul verfügen und zur Anforderung des Roboters ein Telefon benötigt wird, bietet es sich an, die Abstandsmessung über die Bluetooth-Signalstärke des Handys des Benutzers zu realisieren.

Voraussetzungen zur erfolgreichen Abstandsmessung via Bluetooth-Signalstärke sind eine eingeschaltete Bluetooth-Funktion und der Name des Mobiltelefons, welcher in der anfordernden SMS mitgesendet wird. Sind diese Prämissen erfüllt, so kann die Abstandsmessung, wie nachfolgend beschrie-

ben realisiert werden.

Auf dem Smartphone des CampusGuide wird zunächst der *Discovering-Service* gestartet, welcher die Umgebung auf alle in Reichweite befindlichen Bluetooth-Geräte scannt. Alle Namen der gefundenen Geräte werden mit der Kennung des Nutzertelefons verglichen. Findet der *Discovering-Service* die passende Bluetooth-Kennung, wird dieser Service beendet und mit dem Smartphone des Roboters gekoppelt. Bis zur Freigabe des gefundenen Geräts wird permanent die Bluetooth-Signalstärke über den *Received Signal Strength Indication* (RSSI) Wert zwischen den gekoppelten Parteien ermittelt. Über ein Zeitintervall t wird der Median der Messwerte bestimmt und als Maß für die Entfernung zum Benutzer ausgewertet. Überschreitet der gemessene Wert einen Schwellwert oder werden über einen definierten Zeitraum keine Signale empfangen, so gilt der Benutzer als verloren. In diesem Fall könnte CampusGuide seine Fahrt unterbrechen und solange stehen bleiben, bis das Signal wieder detektiert wird.

Durch Messungenauigkeiten können in den gemessenen RSSI-Werten viele Ausreißer beobachtet werden. Aufgrund seiner Robustheit gegenüber extrem abweichenden Werten haben wir uns für den Median zur Mittelung der Messwerte entschieden und betrachten daher nicht das arithmetische Mittel.

4.2.6 Kommunikation mit dem Roboter

In den vorangegangenen Abschnitten werden mehrfach Daten an das Blackboard gesendet oder Daten von diesem empfangen. Hierzu bedarf es einer Kommunikation vom Smartphone zum Roboter und auf dem Roboter selber zum Blackboard. Als Erstes muss also ein Kommunikationskanal zwischen Smartphone und Roboter hergestellt werden. Hierzu kommen mehrere Methoden in Frage: USB, WLAN sowie Bluetooth. Die USB-Variante ist auf den ersten Blick natürlich am besten geeignet, da sie im Gegensatz zu WLAN oder Bluetooth über ein Kabel und nicht über Funk senden und empfangen kann. Dies ist wichtig für eine gute Verbindungsqualität, da hier äußere Störeinflüsse nicht so leicht die Kommunikation beeinflussen können. Wenn man allerdings von einem Android-Smartphone aus über USB kommunizieren will, ist dies keine triviale Aufgabe. Das Betriebssystem verbietet jede Kommunikation über diesen Port. Aufgrund des Benutzersystems, hat der Ausführende der Applikation niemals die erforderlichen Rechte dafür.

Der zweite Ansatz ist WLAN. Diese Methode können wir direkt ausschließen, da CampusGuide mit dem Internet verbunden sein muss. Dies impliziert, dass der Roboter eine ständige Verbindung zum WLAN der Universität oder zum, durch das Smartphone bereitgestellten, Internet haben muss. Hierbei würde sich der Roboter bei der Fahrt an diversen *Accesspoints* anmelden

und könnte somit nicht einem Ad-Hoc Netzwerk des Smartphones beitreten. Zudem wäre es auch nicht möglich zu garantieren, dass Telefon und Roboter zu jeder Zeit mit dem gleichen Netz verbunden sind, um dann über dieses zu kommunizieren.

Dies führt uns zur letzten Methode: Die Bluetooth-Verbindung. Wie schon in der Einführung angesprochen, benutzen Smartphone und Roboter den gleichen Bluetooth-Stack. In unserer Architektur stellt der Roboter den Server für die Kommunikation bereit und das Smartphone ist der Client. Wir haben uns an dieser Stelle für ein zweistufiges Konzept entschieden, wo in der ersten Stufe die Bluetooth Daten empfangen werden und in der Zweiten interpretiert und an das Blackboard weitergeleitet werden. Es wurde also für den Roboter eine Bluetooth-Server Anwendung implementiert, die Daten empfangen kann und diese dann in die gewünschte Richtung weiterleitet. Diese Server-Anwendung wartet auf die Annahme der Kommunikation durch das Smartphone. Die Anwendung auf dem Smartphone sucht genau diesen Dienst auf dem Roboter und kann so eine Kommunikation initiieren. Der Datentransfer selber findet über Nachrichten in einer speziellen Kodierung statt.

Vom Smartphone zum Roboter:
Neue eingehende SMS: *2*Nummer#SMS-Typ#Nachricht#Zeitstempel***
Neue GPS-Position: *1*Latitude#Longitude#Orientierung#Zeitstempel***
Neue Distanz: *3*Wert***

Vom Roboter zum Smartphone:
Neues Passwort: **Passwort***

Bei der Passwort-Nachricht bedarf es keiner erneuten Übertragung der Nummer, da diese intern gespeichert wir, damit weitere Anfragen für CampusGuide abgewiesen werden können.

Jetzt können alle Daten des Smartphones an den Roboter übertragen werden, womit nun lediglich die Anbindung an das Blackboard und die Nachrichteninterpretation fehlen. Die zweite Stufe unseres Konzepts beschreibt genau ein solches Programm. Es empfängt die Daten, interpretiert die Nachrichten und schreibt deren Werte in die dafür vorgesehenen Variablen des Blackboards. Diese Programmstruktur erfüllt all unsere Anforderungen an die Kommunikation zwischen Roboter und Smartphone.

Die Verwendung des Smartphones hat sich für CampusGuide als sehr nützlich und gute Wahl als Mediator zwischen dem Benutzer und der Roboterplattform herausgestellt. Wir sind jetzt in der Lage eine Anforderung

von CampusGuide durch einen Nutzer entgegen zu nehmen, ihm zu antworten und auch um den Abstand zu ihm zu messen. Auf der anderen Seite können wir die empfangenen Daten vom Nutzer sowie Positionskoordinaten an den Roboter weiterleiten und auch wieder empfangen. Diese Funktionalitäten bilden sozusagen die Grundlage um CampusGuide nach außen hin für Anwender nutzbar zu machen.

4.3 Navigation

Der Aspekt Navigation umfasst bei CampusGuide mehrere Teilaspekte. Zuerst muss der Roboter herausfinden, wo genau er sich befindet. In einem Außenareal kann er sich anhand der GPS-Informationen des Handys lokalisieren. Innerhalb eines Gebäudes müssen jedoch Verfahren verwendet werden, die ohne GPS auskommen. Zusätzlich wird für die Planung einer Route eine Karte benötigt. Diese kann, je nach Umgebung, entweder aus Daten der OSM oder lokal vom Roboter erzeugt werden. Ist die Position des Roboters bekannt und liegt eine geeignete Karte vor kann ein Pfad zu einem gegebenen Zielort geplant werden. Während der Fahrt muss CampusGuide Hindernisse erkennen und ihnen ausweichen. Dies gilt sowohl für Objekte, die seinen Weg blockieren, wie z.B. geparkte Autos als auch für Untergründe, die für den Roboter nicht befahrbar sind. Das Problem, die Befahrbarkeit oder *Drivability* eines Untergrunds zu bestimmen wird in Kap. 4.4 näher erläutert.

4.3.1 Lokalisierung

Damit der Roboter überhaupt von einem Ort zu einem anderen fahren kann, ist es natürlich notwendig, dass er jederzeit weiß, wo er sich gerade befindet. Sehr einfach lässt sich dies mit Hilfe von GPS realisieren. Ein Empfänger liest das Signal von mehreren GPS-Satelliten und kann daraus auf mehrere Meter genau seine Position auf der Erdoberfläche bestimmen. Leider sind GPS-Signale innerhalb der meisten Gebäude nicht verfügbar und so greifen wir auf einen Karten basierten Ansatz zurück.

Da in den meisten Fällen keine genaue Karte eines Gebäudes, besonders mit der vorhandenen Einrichtung, zur Verfügung steht, muss diese vorher, mit Hilfe des Laserscanners am Roboter, erstellt werden. Ein Laserscan besteht dabei aus einer Menge von Punkten, die aus einem Winkel und einer Entfernung bestehen. Damit der Roboter aus mehreren Laserscans eine Karte erstellen kann, muss er wissen, wo er sich in dem Gebäude befindet. Damit er sich aber lokalisieren kann, benötigt er eine Karte. Um diese gegenseitige Abhängigkeit zu überwinden, kann der Roboter eine Karte relativ zu seiner eigenen Position erzeugen und gleichzeitig versuchen sich auf dieser lokalisieren. Dabei orientiert er sich an markanten Punkten (Merkmale). Für die Lösung dieses SLAM-Problems existiert mittlerweile eine Vielzahl von Ansätzen unter denen wir uns für eine Implementierung des GMapping genannten Verfahrens entschieden haben [28][29]. Es nutzt einen sogenannten *Rao Blackwellized Partikelfilter* (RBP), um rasterbasierte Karten zu erzeugen. Diese Karten repräsentieren die Umwelt durch eine Menge von Zellen, die entweder belegt (schwarz) oder frei (weiß) sein können. Durch Abstufungen

des Grauwerts wird die Wahrscheinlichkeit, dass eine Zelle frei ist, abgebildet. Jeder Partikel im RBP enthält die vermutete Roboterposition, Merkmale extrahiert aus den Sensordaten des Roboters und eine Vermutung darüber, wie die Karte aussieht. Durch Überleben des passendsten Partikel bestehen nur die Partikel fort, deren Kartenhypothese am besten mit der aktuellen Sensormessung übereinstimmt. Abb. 4.4 zeigt das Ergebnis der GMapping-Implementierung, nach dem der Roboter einmal durch das Gebäude gefahren ist.

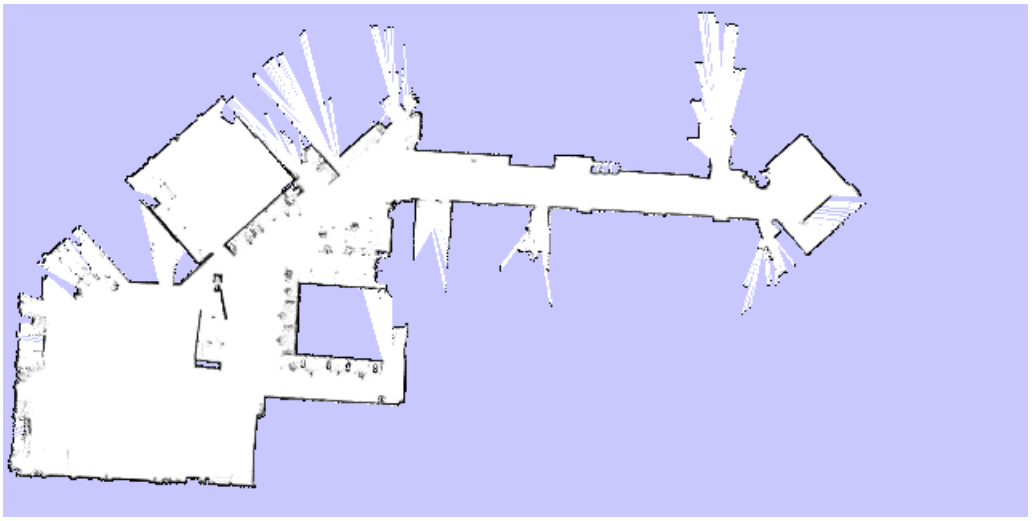


Abbildung 4.4: Rasterkarte unserer Testumgebung, erstellt mit GMapping. Freie Bereiche sind weiß markiert, Zellen die ein Hindernis darstellen schwarz. Alle Zellen über die keine Aussage getroffen werden konnte, sind blau gefärbt.

Die erstellte Karte wird über den Player Treiber *mapfile* in den Player Server geladen und dadurch allen anderen Treibern zur Verfügung gestellt. Zur Lokalisierung nutzen wir den *amcl* Treiber, der im Player-Framework enthalten ist. Dieser bekommt als Eingabe die rasterbasierte Karte, Laserscans sowie die Odometriedaten von CampusGuide und liefert die wahrscheinlichste Position des Roboters zurück. *Adaptive Monte-Carlo Localization* (AMCL) basiert ebenfalls auf Partikelfiltern wobei jeder Partikel für eine mögliche Roboterposition steht. In jedem Aktualisierungsschritt werden die Partikel anhand der Odometrie Informationen bewegt. Danach werden die verschiedenen Partikel anhand der Lasermessung gewichtet; je besser die Messung zu der Position des Partikels passt, desto höher ist die Wahrscheinlichkeit das sich der Roboter dort befindet. Aus dem entstandenen Set wird durch Ziehen mit zurücklegen eine neue Menge von Partikeln gezogen. Dadurch ver-

schwinden die Partikel, deren Position unwahrscheinlich ist und es entsteht eine Häufung von Partikeln dort, wo sich der Roboter am wahrscheinlichsten befindet. Um Ressourcen zu sparen, passt der Algorithmus die verwendete Anzahl der Partikel zur Laufzeit dynamisch an. Je sicherer die Roboterposition ist, desto weniger Partikel befinden sich im Einsatz und umgekehrt.

Die so gewonnene Position des Roboters wird an die lokale Hindernisvermeidung weitergegeben, die diese dann zusammen mit der Zielposition nutzt, um den Roboter an den gewünschten Ort zu bewegen.

4.3.2 Kartenmaterial der OpenStreetMap

Damit sich der Roboter auch in Außenbereichen lokalisieren und einen Weg planen kann, wird eine Karte des gesamten Campus benötigt. Diese kann nicht mittels SLAM erzeugt werden da, bedingt durch die maximale Reichweite des Laser-Entfernungsmessers, nicht genug eindeutige Merkmale gefunden werde können. Um die Navigation mittels des GPS-Moduls des Handys zu ermöglichen, sollten dieser Karte außerdem GPS-Koordinaten zugeordnet werden können. Solche GPS-Daten holen wir automatisch aus der Datenbank der OSM, da diese uns eine lizenzkostenfreie Möglichkeit bietet, an detaillierte Karteninformationen mit hoher Genauigkeit zu gelangen.

4.3.2.1 OSM-Daten automatisch herunterladen

Alle Daten der OSM sind in einer zentralen Datenbank gespeichert. Es gibt verschiedene Wege, um auf diese Daten zuzugreifen:

1. Manuell mittels eines **Clients** wie z.B. JOSM[2]
2. Für einen bestimmten Bereich, direkt über die **OSM-API**

Die OSM bietet mit der OSM-API eine Programmierschnittstelle, mit der wir die benötigten Daten automatisch beziehen können. Somit kann der Roboter immer mit aktuellen Daten arbeiten. Über die OSM-API kann jedoch immer nur ein begrenzter Kartenausschnitt auf einmal heruntergeladen werden. Der OSM-Server liefert derzeit maximal einen Bereich von 0,23 Quadratgrad, was in Deutschland etwa einem Bereich von $50km^2$ entspricht. Dies ist mehr als ausreichend um die Daten für den kompletten Einsatzbereich von CampusGuide mit einem API-Aufruf herunterzuladen.

Die OSM-API basiert auf der *Representational State Transfer* (REST) Architektur. Das heißt, dass jede Anfrage an die API in Form einer URL

[2]http://josm.openstreetmap.de/

gestellt werden kann. Um z.B. die Daten des Campus der TU Dortmund herunterzuladen wird folgender Aufruf verwendet:

.../api/0.6/map?bbox=bbox=<7.40526>,<51.48729>,<7.427>,<51.49885>

Der genaue Ausschnitt, der herunterladen werden soll, ist durch eine *Bounding Box* aus 4 GPS-Koordinaten eingegrenzt. Für den Fall, dass sich der Operationsbereich von CampusGuide auf dem Gelände der TU ändert, oder er an einem anderen Ort eingesetzt werden soll, müssen nur diese Werte modifiziert werden.

4.3.2.2 GPS-Wegpunkte extrahieren

Die OSM liefert auf Anfragen durch die API Daten in Form einer XML-Datei zurück. Wir haben ein Programm entwickelt, das automatisch die Wege aus einer solchen XML-Datei extrahiert und aufbereitet. Eine solche von der OSM bezogene Datei enthält Informationen wie z.B. Punkte mit dazugehörigen GPS-Koordinaten (*Nodes*), Wege, Relationen und *Tags*. Diese werden für die Darstellung von Objekten auf der Karte verwendet. Ein Gebäude wird dabei z.B. durch mehrere verbundene Punkte repräsentiert, denen der gemeinsame *Tag building = yes* zugewiesen wird. Von den erhaltenen Informationen, sind für den Roboter nur die Punkte von Interesse, die Wege repräsentieren. Wege werden in der OSM durch eine Menge verbundener Wegpunkte repräsentiert, die mit dem Tag *Highway* versehen sind. Der Wert der dem *Highway-Tag* zugeordnet wurde gibt an, um welche Art Weg es sich handelt (z.B. Autobahn, Fußweg). Anhand dieser *Tags* werden genau die Wege aus der Datei extrahiert, die für CampusGuide prinzipiell befahrbar sind. Aus diesen Wegen und ihren Knotenpunkten wird ein Graph erzeugt, auf dem die Pfadplanung von CampusGuide durchgeführt wird.

4.3.3 Pfadplanung

Die von uns verwendete Roboterplattform ist nicht dazu geeignet, sich autonom auf dem Campus zu bewegen. So kann sie z.B. keine absteigenden Treppen erkennen, nicht über Wiesen fahren und auch keine zu starken Steigungen bewältigen. Damit der Roboter trotzdem sicher und vor allem schnell von seiner aktuellen Position ein anderes Ziel erreichen kann, bedarf es eines Mechanismus, der ihm zu jedem Zeitpunkt den optimalen Weg bereitstellt. Der optimale Weg ist dabei der kürzeste Weg, der mit höchster Wahrscheinlichkeit von unserer Roboterplattform befahrbar ist. Da wir über eine gute

lokale Hindernisvermeidung verfügen, muss der Pfad nicht sehr genau angegeben werden. Es ist ausreichend, eine Liste von Koordinaten von allen markanten Punkten auszugeben. Markante Punkte stellen neben dem Ziel vor allem Kreuzungen dar, an denen abgebogen werden muss. Um schnell von A nach B zu kommen, muss der Roboter nur noch die Liste der Wegpunkte sukzessive abarbeiten. Als Struktur zur Implementierung der Pfadplanung bieten sich aus folgenden Gründen Graphen an:

1. Aus Graphen erhält man sehr einfach Listen von Knoten (die Wegpunkte repräsentieren)

2. Sie sind gut erforscht und es steht eine breite Auswahl an effizienten Algorithmen zur Verfügung

3. Es gibt frei verfügbare Implementierungen

Zur Realisierung kommt die LEMON Graph Library zum Einsatz [30]. LEMON steht für *Library for Efficient Modeling and Optimization in Networks* und ist eine in C++ geschriebene und frei verfügbare Bibliothek, die eine Vielzahl von häufig verwendeten Graphenstrukturen und -algorithmen effizient implementiert.

Der Graph wird aus OpenStreetMap Daten aufgebaut. Jeder Knoten repräsentiert eine OSM *Node*, welche aus einer eindeutigen ID und GPS-Koordinaten besteht. Eine Kante wird zwischen zwei Knoten genau dann eingefügt wenn, es einen OSM-Weg zwischen zwei *Nodes* gibt, der laut Annotation auch von Fahrrädern befahren werden kann. So wird sicher gestellt, dass jeder Weg der durch den Graphen gefunden wird, weder Treppen noch Wiese enthält. Als Kantengewicht wird die Entfernung zwischen den beiden GPS-Koordinaten verwendet. Das ist ausreichend, da in den OSM-Daten zwei *Nodes* immer nur direkt verbunden sind.

Natürlich wird es selten vorkommen, dass der Start- bzw. der Zielpunkt genau auf einem Knoten des Graphen liegt. Dies stellt aber kein Problem dar. Zunächst werden die Knoten ermittelt, die dem Startpunkt und dem Endpunkt am nächsten liegen. Zwischen diesen beiden Knoten wird dann mit dem Dijkstra-Algorithmus [31] der kürzeste Pfad ermittelt. Um den kompletten Pfad zu erhalten, muss an die resultierende Liste nur noch der Zielpunkt als letzter Wegpunkt angehängt werden.

Die gesamte Implementierung wurde als Player-Plugin realisiert. Dabei wurde das *planner*-Interface komplett so implementiert, dass der Treiber problemlos mit jedem anderen Treiber aus dem Player-Framework, der dieses Interface nutzt, funktioniert.

4.3.4 Hindernisvermeidung

Hat sich CampusGuide lokalisiert und eine Pfad zum Zielpunkt bestimmt, so kann es trotzdem sein, dass der er einem Hindernis begegnet. Dabei kann es sich z.B. um eine Laterne, ein geparktes Auto, oder ein beliebiges anderes Objekt handeln, das nicht in der OSM verzeichnet ist und deshalb vom Pfadplaner nicht berücksichtigt werden konnte. Insbesondere bewegliche Hindernisse, wie z.B. Menschen, müssen erkannt werden. Damit CampusGuide keine Gefahr für Passanten darstellt, bedarf es einer robusten Methode, die erkennt,wenn etwas oder jemand den Weg versperrt. Wurde ein Hindernis entdeckt, muss ein Ausweichkurs um das Hindernis berechnet werden, bevor CampusGuide auf seiner geplanten Route weiterfahren kann.

Wir haben uns entschieden auf die Implementierung des VFH+ Algorithmus zurückzugreifen, die bereits im Player-Framework zur Verfügung steht. Der Algorithmus ist ein weit verbreiteter Ansatz und hat sich als robust und effizient erwiesen. VFH wurde 1991 von Borenstein und Koren vorgestellt; 1998 wurde er durch Ulrich und Borenstein verbessert und in VFH+ umbenannt [32]. Aus den Daten des Laserscanners erzeugt der Algorithmus ein polar Histogramm um die Position des Roboters herum. Täler in diesem Histogramm werden als Hindernisse interpretiert und es wird ein Set von möglichen Fahrtrichtungen generiert. Aus diesem Set wird die Richtung gewählt, die der Zielrichtung am nächsten ist. Die Implementierung im Player Framework erhält dafür einen Zielpunkt und berechnet daraus Steuerbefehle, die den Roboter kollisionsfrei zum Ziel bringen sollen. Dazu muss der *vfh* Treiber neben den Laserdaten auch Zugriff auf die aktuelle Roboterposition haben. Zur Kommunikation zwischen dem *vfh* Treiber und der Roboter-Hardware wird das *position2d*-Interface verwendet. Es bietet sowohl die Möglichkeit die kartesischen Positionsdaten des Roboters auszulesen, als auch Steuerbefehle von anderen Treibern anzunehmen. Die Implementierung von VFH im Player-Framework erlaubt nur den Zugriff auf ein *position2d*-Interface. Für unsere Implementierung stellt das ein Problem dar, denn wir verwenden zur Lokalisation und Steuerung zwei getrennte Module. Der Treiber der Firma MetraLabs nimmt die Steuerbefehle an und der *amcl* Treiber liefert die aktuelle Position. Zwar kann der MetraLabs Treiber auch die Roboterposition anhand von Odometriedaten liefern, doch hat diese Methode einen entscheidenden Nachteil: Durch unebenen oder rutschigen Boden stimmt die durch die an den Getrieben angebrachten Encoderscheiben bestimmte Bewegung nicht mit der tatsächlichen Roboterbewegung überein. Diese Fehler summieren sich sehr schnell auf und nach kurzer Zeit wäre die gelieferte Position für unsere Zwecke zu ungenau. Wir haben deswegen den *vfh* Treiber aus dem Player-Framework um ein weiteres *position2d*-Interface erweitert. Den Trei-

ber haben wir dahin gehend modifiziert, dass er die Position von dem einem Modul bezieht und die Steuerbefehle an das andere Modul sendet.

Damit ist die aus drei Ebenen bestehende Architektur komplett. Der globale Planer sendet seinen gewünschten Pfad an die lokale Hindernisvermeidung. Diese versucht dem Pfad so gut wie möglich zu folgen, bezieht dabei aber noch Sensordaten mit ein, um unerwarteten Hindernissen ausweichen zu können. Die Hindernisvermeidung generiert die Steuerbefehle, die dann an die Motoren des Roboters gesendet werden.

4.4 Befahrbarkeit

Als Befahrbarkeit oder *Drivability* wird die Eigenschaft eines Untergrundes bezeichnet, die angibt, ob dieser Untergrund von einem Roboter befahren werden kann oder nicht. Dies ist natürlich stark von dem Modell und der Ausstattung des Roboters abhängig. Da der SCITOS G5 für den Einsatz in Innenräumen konstruiert wurde, kommt nur eine kleine Menge an Untergründen für CampusGuide überhaupt in Frage. Die nicht befahrbaren Untergründe soll der Roboter selbstständig als Hindernisse erkennen und umfahren.

Die Befahrbarkeit eines Untergrunds kann auf mehrere Arten bestimmt werden. Steht ein geeigneter Sensor zur Verfügung, kann die Struktur des Bodens direkt betrachtet werden. Der Roboter den Morales et al. [12] für die *Tsukuba Challenge* verwendet haben, verfügt z.B. über zwei diagonal auf den Boden gerichtete Laser-Entfernungsmesser, mit denen Erhöhungen, Treppenstufen und andere Unebenheiten erkannt werden können. Da uns ein solcher Sensor nicht zur Verfügung steht, haben wir uns entschieden, den Untergrund mit Hilfe von Kamerabildern zu klassifizieren. Dafür werden Merkmale aus den Bildern extrahiert und mit Hilfe eines künstlichen neuronalen Netzes ausgewertet.

Für dieses Verfahren verwenden wir nur die Kamera, die direkt in Fahrrichtung des Roboters auf den Boden gerichtet ist (Siehe Abb. 3.1, Boden-Kamera). Die obere Kamera ist für diese Aufgabe nicht geeignet, da diese drehbar gelagert ist und sich während der Fahrt verstellen kann. Dadurch ist es nicht möglich sichere Rückschlüsse von einem Punkt des Bildes auf eine Position vor dem Roboter zu ziehen.

Das Modul ist als *Pipeline* aufgebaut, die in regelmäßigen Abständen Bilder der Kamera bezieht, Hindernisse detektiert und ihre Ergebnisse an das Navigations-Modul weitergibt.

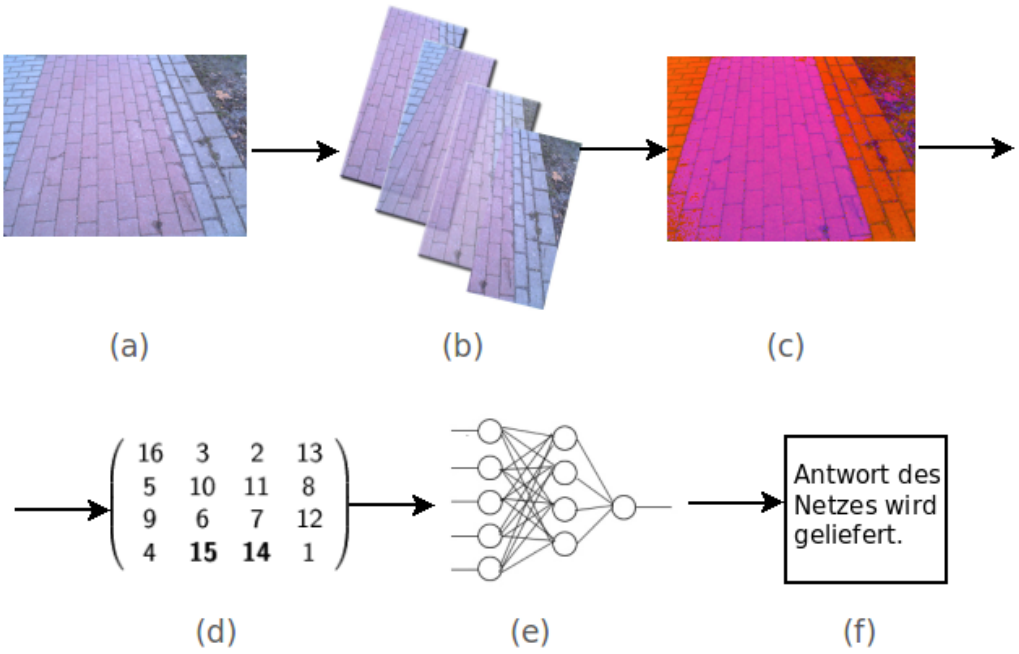


Abbildung 4.5: Pipeline zur Befahrbarkeitserkennung. Von links nach rechts: (a) Aufnahme der Kamerabilder; (b) Einteilung der Bilder in Segmente; (c) Umrechnung der Segmente in den HSV-Farbraum; (d) Erstellung von Merkmalsvektoren über die Bildsegmente; (e) Eingabe der Merkmalsvektoren in das neuronale Netz; (f) Entscheidung des neuronalen Netzes zur Befahrbarkeit des Bildsegmentes

4.4.1 Pipeline

Der Ablauf der Befahrbarkeits-Erkennung folgt der in Abbildung 4.5 dargestellten Pipeline. Das von der Kamera gelieferte Bild wird nicht im Ganzen betrachtet, sondern in kleine Segmente unterteilt. Für jedes dieser Teilbilder werden Merkmale extrahiert und ein Merkmalsvektor erstellt, der die Eingabe für das nachgeschaltete neuronale Netz bildet. Fällt die Antwort des Netzes für den eben betrachteten Bildausschnitt negativ (*nicht befahrbar*) aus, wird dieser Bereich in den Laserdaten als Hindernis eingesetzt.

Um die *Pipeline* in Funktion zu testen haben wir zuerst einen Erkenner implementiert, der nur auf Basis von Farbwerten arbeitet. Dieser betrachtet das komplette Bild und verzichtet auf die Merkmalsextraktion aus den Teilbildern.

4.4.2 Farberkenner

Der von uns implementierte Farberkenner kann auf einen bestimmten Farbwert trainiert werden. Dieser wird als Hindernis erkannt und als virtuelles Objekt in die Daten des Laser-Entfernungsmessers geschrieben.

Um den Erkenner zu trainieren, haben wir aus Bildern der Roboterkamera die Bereiche extrahiert, die als Hindernis klassifiziert werden sollen. Auf Basis der extrahierten Farbwerte haben wir ein Histogramm im RGB Farbraum erzeugt, das den Farbbereich der Hindernisse beschreibt. Diesen Bereich haben wir mit zwei quadratischen Funktionen approximiert, die als Eingabe den Farbwert eines zu klassifizierenden Pixel erhalten. Zu einem Eingabebild kann mit diesem Verfahren im laufenden Betrieb eine Karte der Befahrbarkeit dieses Bildes erzeugt werden.

Die Karte muss an das Navigationsmodul weitergegeben und in die Hindernisvermeidung einbezogen werden. Da die verwendete Kamera auf dem Roboter in fester Position und Ausrichtung installiert ist, kann aus einer Position im Bild auf einen Punkt auf dem Boden vor dem Roboter geschlossen werden. Dies gilt, da wir davon ausgehen, dass sich der Roboter auf ebenen Flächen bewegt. Für jedes Pixel, das ein Hindernis repräsentiert, kann also ein Winkel und ein Abstand vom Roboter berechnet werden. Diese Daten werden mit denen des Laser-Entfernungsmessers kombiniert, dessen Werte ebenfalls als Entfernungen für eine Menge an Winkeln bestimmt werden. Es wird für alle Winkel das Minimum über den Abstand, den der Laser-Entfernungsmesser liefert und den vom Farberkenner bestimmten Abstand gebildet. Dadurch wird ein vom Farberkenner detektiertes Hindernis als „virtuelles Hindernis“ für den Laser modelliert und dadurch automatisch von der Hindernisvermeidung berücksichtigt.

Da der Farberkenner auf einen bestimmten Farbwert trainiert werden muss, kann er nur sinnvoll in unserer Testumgebung verwendet werden. Um eine Befahrbarkeitserkennung in Außenarealen zu realisieren, müssen wir auf ein breiteres Farbspektrum reagieren. Daher ist der Farberkenner für das angestrebte Szenario nicht ausreichend. Zusätzlich möchten wir strukturelle Merkmale in die Erkennung einbeziehen. Dazu kommt ein künstliches neuronales Netz zum Einsatz.

4.4.3 Annotation

Zum Training eines künstlichen neuronalen Netzes wird eine Datenbasis benötigt, von der das Netz lernen kann. In unserem Fall mussten wir für einen Satz Bilder von Hand festlegen, ob der Untergrund für den Roboter befahrbar ist. Da die Menge der zu annotierenden Bilder sehr groß war und eine pixel-

genaue Einfärbung damit zu aufwändig, haben wir ein Programm entwickelt, das diese Annotation vereinfacht. Ein eingegebenes Bild sollte nach der Verarbeitung in Segmente aufgeteilt sein, die mit einem passenden Label für den Untergrund den sie zeigen versehen sind. Wir haben uns für die Codierung in Farbwerten entschieden (grün: befahrbar, blau: vielleicht befahrbar, rot: nicht befahrbar), da sich diese leicht in das Ausgangsbild einblenden lassen.

Zum Segmentieren der Bilder haben wir zwei Ansätze verfolgt. Zum einen haben wir versucht mit Hilfe des *Canny Edge Detectors* Trennlinien zwischen einzelnen Segmenten zu bestimmen. Dieses Verfahren tendierte jedoch dazu, auch Flächen zusammenfassen, die nicht zusammengehören. Ein Beispiel ist in Abbildung 4.6 zu sehen. Der zweite Ansatz basiert auf dem *Watershed* Algorithmus. Dieser zerlegt anhand von vorgegebenen Startwerten ein Grauwertbild in verschiedene Segmente. Mit gut gewählten Startwerten konnten wir so eine für unsere Zwecke ausreichend genaue Segmentierung erreichen.

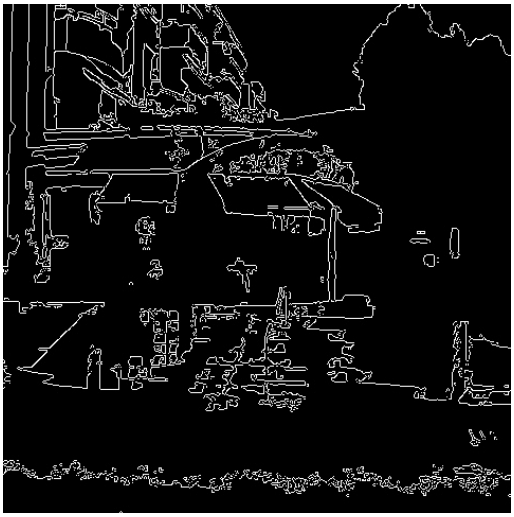
Der Benutzer muss die einzelnen Regionen des Bildes von Hand mit Markern versehen (siehe Abb. 4.6c). Diese Marker bilden die Startwerte für den *Watershed* Algorithmus, der eine zusammenhängende Fläche für jeden gegebenen Startwert erzeugt. Jedem gefundenen Segment kann daraufhin ein Label zugewiesen werden das die Befahrbarkeit des Untergrunds repräsentiert (siehe Abb. 4.7c).

Mit diesem Tool kann für jede Bilddatei eine Karte der Befahrbarkeit erstellt werden, auf deren Basis ein künstliches neuronales Netz trainiert und validiert werden kann. Wir verwenden einen so vorbereiteten Datensatz ebenfalls zur Evaluation des Befahrbarkeits-Moduls.

4.4.4 Merkmalsextraktion

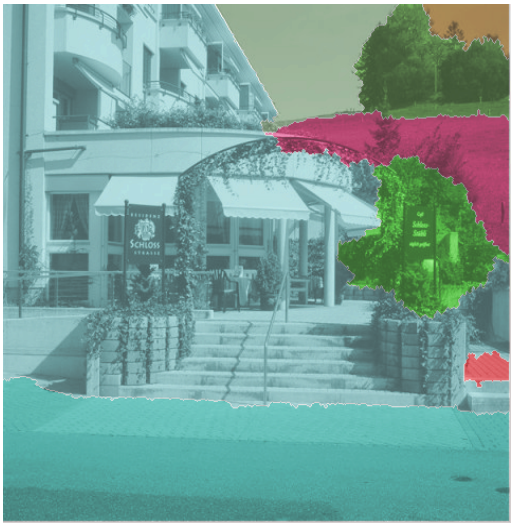
Zur Erkennung der Befahrbarkeit eines Untergrunds werden aus dem Kamerabild Merkmale extrahiert, welche die Eingabe für das neuronale Netz darstellen. Hierbei haben wir zwei Ansätze verfolgt. Zum einen erstellen wir ein Farbhistogramm und entscheiden über die Befahrbarkeit eines Bildausschnitts anhand der Farbe des Untergrunds und zum anderen arbeiten wir mit Gaborfiltern zur Kantenerkennung, um den Untergrund anhand seiner Struktur zu klassifizieren. Zum Training des neuronalen Netzes werden diese Merkmalsvektoren mit den Annotationen aus 4.4.3 verknüpft, so dass für jeden Merkmalsvektor bekannt ist, ob er befahrbaren oder nicht befahrbaren Untergrund repräsentiert. Im laufenden Betrieb soll das Netz einen Merkmalsvektor als Eingabe erhalten und von diesem auf die Befahrbarkeit schließen.

(a) Ausgangsbild



(b) Mit dem *Canny Edge Detector* gefundene Kanten. Bäume, Wiese und Gebäude im oberen rechten Teil des Bildes werden zu einer Fläche zusammengefasst. Ebenso werden im Zentrum des Bildes Pflastersteine und Treppenstufen nicht getrennt.

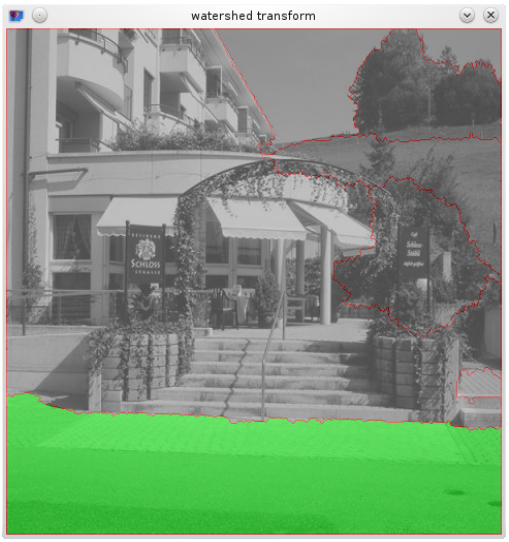
(c) Bild versehen mit acht Markern für den *Watershed* Algorithmus (weiße Linien).



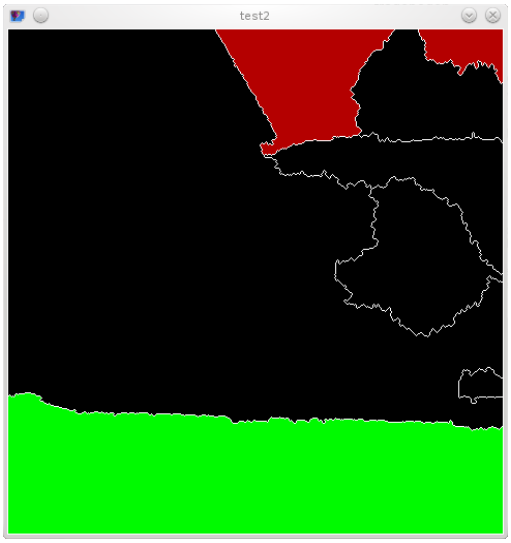
(d) Mithilfe des *Watershed* Algorithmus segmentiertes Bild. Die verschiedenen Flächen sind klar getrennt. Das Ergebnis hängt jedoch stark von der Wahl der Marker ab.

Abbildung 4.6: Vergleich der Beiden Verfahren zur Segmentierung der Eingabebilder.

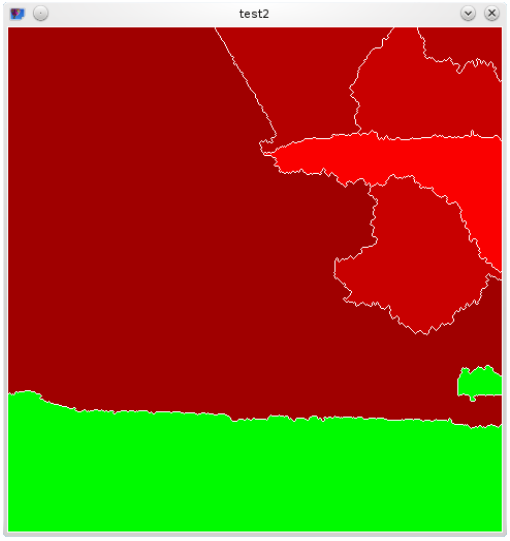
(a) Nach der Anwendung des *Watershed* Algorithmus vorliegendes Bild. In diesem können Segmente durch klicke ausgewählt werden.



(b) Das grün markierte Segment wurde ausgewählt. Diesem kann ein Untergrund bzw. ein Label zugewiesen werden.



(c) Die Maske der zugewiesenen Untergründe. Drei Segmente wurden bereits bearbeitet. Dem unteren Segmente wurde das Label „Pflastersteine“, den beiden oberen das Label „Himmel“ zugewiesen.



(d) Die vollständige Maske der Untergründe.

Abbildung 4.7: Zuweisung von Labels an die einzelnen Segmente eines Bildes.

4.4.4.1 Farbhistogramme

Zur Betrachtung der Farben eines Untergrunds benötigen wir einen passenden Farbraum. Der Ansatz den nativen RGB-Farbraum der verwendeten Kamera zu benutzen schied aus, da hierbei die Umgebungshelligkeit eine zu große Rolle spielt und somit die Ergebnisse zu verschiedenen Tageszeiten und bei unterschiedlichen Witterungen starke Streuungen aufwiesen. Wir entschieden uns für einen Farbraum, in dem die Helligkeit in einem einzelnen Kanal dargestellt wird. Zur Auswahl stehen hier die Farbräume: *hue-saturation-value* (HSV), *normalized*-RG (NRG) und Lab. Für die Klassifikation kann man diese dann auf die Farbkanäle beschränken um ein helligkeitsunabhängiges Ergebnis zu erzielen und so mögliche Probleme mit dem Umgebungslicht zu minimieren. Wir betrachten zur Erstellung des Farbhistogramms einen Frame, der von der Kamera geliefert wird. Diesen unterteilen wir mittels einem sich verschiebenden Fensters (*sliding window*) und klassifizieren jedes Kamerabild Fenster für Fenster. Dieser zu betrachtende Ausschnitt (*region of interest* / ROI) wird in den gewählten Farbraum überführt und analysiert. Es werden je zehn Behälter (*bins*) für beide Farbkomponenten verwendet, um das Farbhistogramm zu erstellen. Diese unterscheiden sich von Farbraum zu Farbraum, wobei z. B. bei HSV die Kanäle Farbton (*hue*) und Sättigung (*saturation*) verwendet werden. Die zehn konkatenierten Zeilenvektoren dieser 10×10 Matrix sind der Eingabevektor für das vorher trainierte neuronale Netz.

4.4.4.2 Gabor

Da Farbe nicht ausreichend ist, um eine möglichst genaue Klassifikation des Untergrunds zu ermöglichen, haben wir zusätzlich ein Verfahren für die Extraktion struktureller Merkmale implementiert, welches z. B. Muster von Pflastersteinen von denen, einer Wiese unterscheiden kann. Wir wenden 2D-Gaborfilter [33] mit acht Orientierungen und vier verschiedenen Frequenzen an, um so möglichst alle Kanten im Bild zu finden.

Hierbei wird das Bild in einem bestimmten Bereich mit einem durch 4.1 definierten Gaborfilter gefaltet. Die Antwort des Filters gibt an, wie gut die Strukturen im Bild mit denen des Filters übereinstimmen.

$$f(x,y) = e^{-\pi[(x-x_0)^2a^2+(y-y_0)^2b^2]}e^{-2\pi i[u_0(x-x_0)+v_0(y-y_0)]}, \tag{4.1}$$

$$F(u,v) = e^{-\pi[\frac{(u-u_0)^2}{a^2}+\frac{(v-v_0)^2}{b^2}]}e^{-2\pi i[x_0(u-u_0)+y_0(v-v_0)]} \tag{4.2}$$

Hierbei ist $f(x,y)$ das Produkt einer elliptischen Gaußfunktion mit dem Mittelpunkt (x_0, y_0) und einer komplexen Exponentialfunktion, die die har-

monische Modulation mit der Ortsfrequenz $\sqrt{u_0^2 + v_0^2}$ und der Orientierung $\arctan \frac{v_0}{u_0}$ darstellt. Das Seitenverhältnis der Gaußfunktion ist hierbei durch b/a definiert. $F(u, v)$ ist die 2D-Fouriertransformation von Gleichung 4.1.

(a) diagonale Orientierung, Testfrequenz 1 (b) horizontale Orientierung, Testfrequenz 1 (c) Testbild (d) horizontale Orientierung, Testfrequenz 2 (e) diagonale Orientierung, Testfrequenz 2

Abbildung 4.8: Beispielantworten der Gaborfilter bei Rasenuntergund

(a) diagonale Orientierung, Testfrequenz 1 (b) horizontale Orientierung, Testfrequenz 1 (c) Testbild (d) horizontale Orientierung, Testfrequenz 2 (e) diagonale Orientierung, Testfrequenz 2

Abbildung 4.9: Beispielantworten der Gaborfilter bei Pflastersteinen

Durch die Verwendung verschiedener Frequenzen fallen die Antworten der Filter jeweils bei entsprechend feinen Untergrundstrukturen stärker oder schwächer aus. Eine starke Antwort ist hier durch einen hohen Weißanteil im Bild repräsentiert. Ein Stück Wiese liefert z. B. durch die vielen Grashalme, die in allen möglichen Richtungen liegen, eine relativ homogene Antwort (siehe Abb. 4.8). Aufgrund der unruhigen Struktur des Rasens ist die Antwort der Filter bei allen Orientierungen und Frequenzen ähnlich. Im Gegensatz dazu wird das Ergebnis nach dem Filtern eines Bildes mit Pflastersteinen (siehe Abb. 4.9) nicht bei jeder Orientierung gleich aussehen. Wenn die Orientierung horizontal verläuft, liegt der Filter genau auf den horizontalen Fugen im Weg und im Ergebnis ist ein deutlich höherer Weißanteil als bei einer, zum Bild, diagonalen Orientierung. Ähnlich verhält es sich bei den Frequenzen, da auch hier die Steine eine sehr homogene Oberfläche haben, sodass Antworten der Filter hauptsächlich bei niedrigen Frequenzen diesen hohen Weißanteil aufweisen. Der Grund hierfür ist das niederfrequente Auftreten von Fugen.

Aus diesen Erkenntnissen haben wir ein Merkmal generiert, welches ein Histogramm über die Filterantworten ist. Die Anzahl der *bins* ist definiert über die Anzahl der Frequenzen und Orientierungen und die *bins* beinhalten die Summe aller Farbwerte der Antwort des Gaborfilters bei gegebenen Parameter.

4.4.5 Klassifikation

Im laufenden Betrieb werden die von der Kamera aufgenommenen Bilder mit einem *sliding window* abgetastet. Aus diesen Teilbildern werden Merkmalsvektoren durch die in Kapitel 4.4.4.1 und in Kapitel 4.4.4.2 beschriebenen Verfahren extrahiert. Die Vektoren werden durch ein neuronales Netz klassifiziert, wodurch eine Karte der Befahrbarkeit des Bildes entsteht. Die so detektierten Hindernisse werden, analog zu dem in 4.4.2 beschriebenen Verfahren, in die Daten des Laser-Entfernungsmessers eingebracht. Dadurch werden die als nicht befahrbar klassifizierten Untergründe automatisch von der Hindernisvermeidung beachtet.

4.5 Mensch-Maschine-Interaktion

Die Mensch-Maschine-Interaktion (MMI) umfasst sowohl die passive als auch die aktive Interaktion mit dem Benutzer. In unserem Projekt erfolgt die passive Interaktion durch einen Personendetektor, der dazu verwendet wird, mögliche Nutzer mit Hilfe des Laser-Scanners, sowie der beweglichen Kamera, zu finden. Die aktive Interaktion mit dem Roboter erfolgt durch den Touchscreen, über den eine grafische Oberfläche (GUI) bedient wird. Des Weiteren kommt ein grafisches Emotionssystem zum Einsatz, das dazu dient, den Roboter menschlicher wirken zu lassen, und den Benutzern so den Zugang zum Roboter zu erleichtern.

4.5.1 Personendetektion

Sobald der Roboter an seiner Einsatzposition angekommen ist, ist das nächste zu erreichende Ziel die Identifikation des Benutzers, der ihn gerufen hat. Zu diesem Zweck wird ein Beinpaardetektor eingesetzt, der die Umgebung nach potenziellen Menschen absucht, und dessen Ergebnisse danach von einem Gesichtsdetektor verifiziert oder abgelehnt werden. Wird auf diese Art eine Person gefunden, so fährt der Roboter zu dieser und bleibt etwa einen Meter vor ihr stehen. Der Nutzer kann sich mit dem per SMS erhaltenen Passwort über die grafische Oberfläche anmelden.

Bei der Frage nach dem Aufbau eines Personendetektors gibt es zwei Problemstellungen. Zum einen soll die Detektion möglichst schnell funktionieren, zum anderen soll ein Ziel angesteuert werden, bei dem es sich tatsächlich um eine Person handelt. Der verwendete HoG-Gesichtsdetektor findet zwar mit hoher Wahrscheinlichkeit Personen, die auf einem Bild zu sehen sind, ist dabei jedoch relativ langsam. Da ein komplettes Absuchen der Umgebung sehr zeitaufwändig wäre, erfolgt eine Vorauswahl von möglichen interessanten Positionen, an denen sich Menschen befinden könnten, durch den Beinpaardetektor. Dieser liefert sehr schnell gute Ergebnisse, die dann vom Gesichtsdetektor verifiziert werden können.

4.5.1.1 Ablauf der Personendetektion

Die Personendetektion startet, sobald der Roboter am Ziel angekommen ist. Zu diesem Zweck wird zuerst der Bereich vor dem Roboter mittels des Lasers in Kniehöhe gescannt. Mit Hilfe der erhaltenen Daten wird nach drei Arten von Objekten (Beinpaaren, einzelnen Beinen und Röcken) gesucht, um die Positionen potenzieller Personen zu ermitteln. Diese werden zuerst nach ihrer Art und dann nach dem minimalen Abstand zum Roboter sortiert, so dass

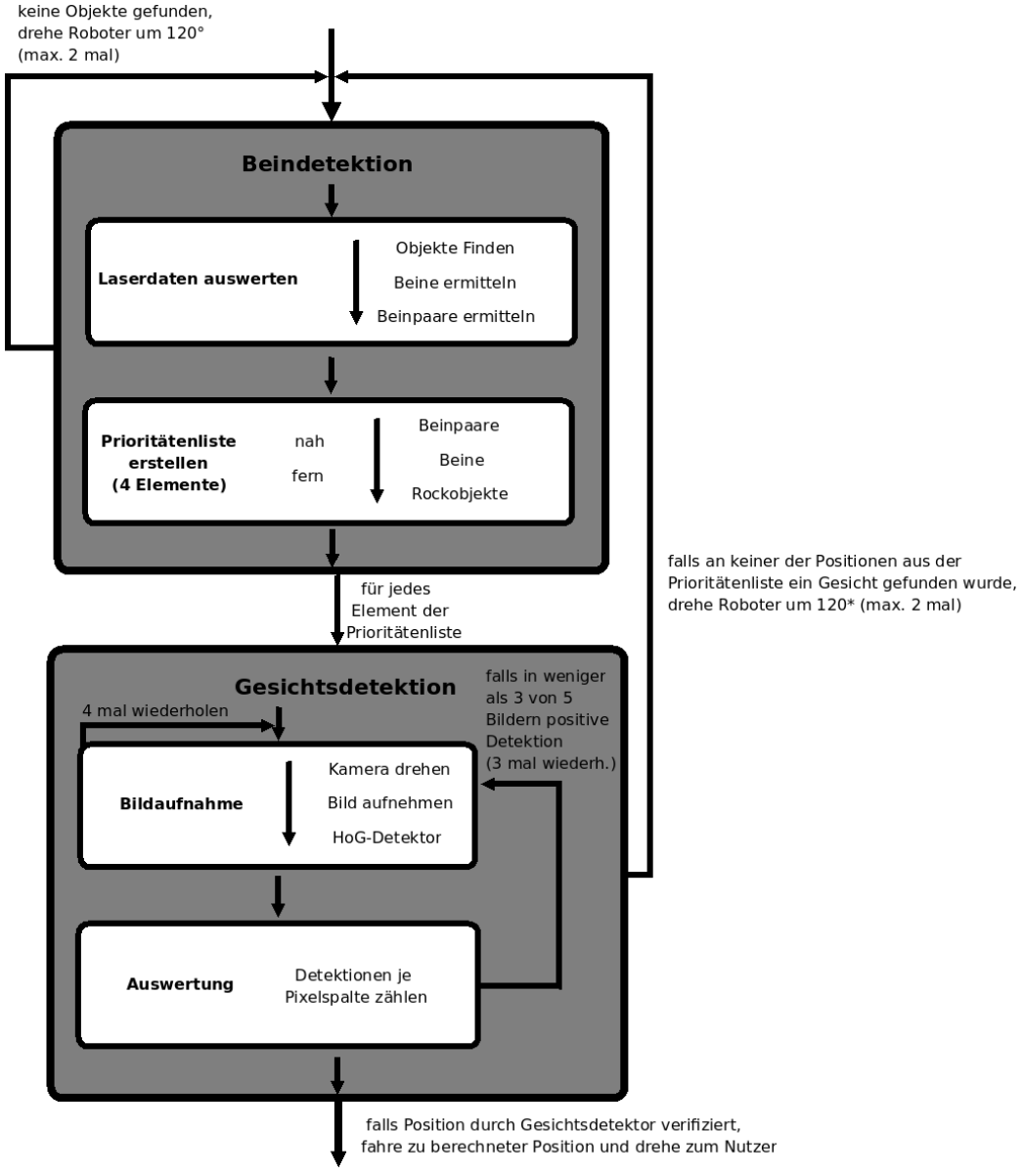


Abbildung 4.10: Ablauf der Personendetektion

detektierte Beinpaare, die nah am Roboter sind die höchste Priorität haben, und Röcke, die sehr weit entfernt sind, die niedrigste Priorität genießen. Von den so ermittelten möglichen Personen werden die vier höchst priorisierten Positionen nacheinander vom Gesichtsdetektor untersucht, wobei der Prozess abbricht, sobald eine Verifikation erfolgt ist. Die Gesichtsdetektion erfolgt auf Bildern, die von der beweglichen Kamera aufgenommen werden. Zu diesem Zweck werden drei nacheinander aufgenommene Bilder betrachtet. Falls auf zwei dieser Bilder ein Gesicht an derselben Position detektiert wird, so wird dies als korrekte Detektion gesehen und der Roboter fährt zur vorher ermittelten Position und dreht sich zum Benutzer, so dass dieser sich über den Touchscreen anmelden kann. Der Aufbau des Systems ist in Abbildung 4.10 dargestellt.

4.5.1.2 Beinpaardetektor

Die Grundidee des Beinpaardetektors ist es mittels des Laser-Scanners Objekte zu finden, die von der Form her Beinpaare sein könnten. Grundlegende Ideen, wie die Ermittlung der Form eines Beines, basieren dabei auf dem Beindetektor der Projektgruppe Partybot[3]. Der Laser-Scanner liefert für die Beindetektion zwei Informationen: einen Winkel und eine Entfernung, aus denen für jeden gescannten Punkt eine Koordinate berechnet wird. Zudem können diese Daten verwendet werden, um Beine zu ermitteln. Dies geschieht in mehreren Schritten.

Zuerst ist es wichtig einzelne Objekte voneinander zu unterscheiden. Zu diesem Zweck werden im ersten Schritt die Entfernungen benachbarter Laserpunkte miteinander verglichen, wobei nur Objekte betrachtet werden, die sich höchstens 5 m vom Roboter entfernt befinden. Um aus den Laserdaten einzelne Objekte zu extrahieren, wird der Abstand zum Roboter von zwei benachbarten Laserpunkten verwendet. Wenn dieser mehr als 23 cm beträgt, wird dies so interpretiert, dass beide Laserpunkte zu verschiedenen Objekten gehören und diese entsprechend indiziert. Somit gilt für jeden Laserpunkt, der demselben Objekt zugeordnet wurde, dass der Abstand zum Roboter im Vergleich zu seinem Nachbarn höchstens 23 cm mehr beträgt.

Für die so unterteilen Objekte wird im zweiten Schritt jeweils ermittelt, ob es sich dabei um ein Bein handeln könnte. Hierzu werden drei Kriterien verwendet, die alle erfüllt sein müssen: Die Beinbreite, die Beintiefe sowie die Beinform. Dabei entspricht die Beinbreite dem Abstand zwischen dem ersten und dem letzten Laserpunkt, die demselben Objekt zugeordnet werden. Dabei werden alle Laserpunkte betrachtet, die einem Objekt zugeordnet

[3]https://eldorado.tu-dortmund.de/bitstream/2003/26365/1/PG525_Endbericht.pdf

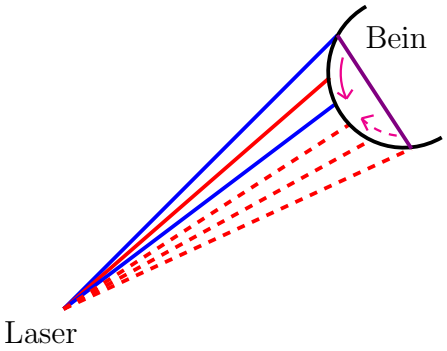


Abbildung 4.11: Beinerkennung (Breite, Tiefe, Zählrichtung für Form(durchgezogen = gegen Uhrzeiger; gestrichelt = mit Uhrzeiger))

werden. Von diesen werden die Laserpunkte mit der zum Roboter kürzesten und weitesten Entfernung voneinander subtrahiert. Das Ergebnis wird in Relation zur Beinbreite gesetzt. Dabei muss die Beintiefe in einem festgelegten Bereich im Vergleich zur Beinbreite liegen, damit dieses Kriterium erfüllt ist. Als drittes Kriterium wird die Beinform betrachtet. Hierfür wird der Verlauf der Laserpunkte verfolgt. Mit Hilfe von zwei Zeigern, die von den Außenpunkten des Objektes zum Mittelpunkt geführt werden, wird ermittelt, ob das Objekt eine leichte Wölbung hat. Sämtliche zuvor ermittelten Objekte, die diese drei Kriterien erfüllen werden als Beine erkannt. Für alle Kombinationen der ermittelten Beine wird als nächstes die Distanz zueinander ermittelt. Liegt diese unter einem Maximalwert, so wird die jeweilige Kombination von Beinen als Beinpaar klassifiziert. Zuletzt werden aus den nicht als Bein erkannten Objekten alle Objekte als Rock klassifiziert die eine bestimmte Breite haben. Dafür muss ein Objekt mindestens eine Breite von zweifacher minimaler Beinbreite haben, sodass auch zwei Beine, die direkt nebeneinander stehen und somit nicht als einzelne Beine erkannt werden können, in die Kategorie Rock eingeordnet werden. Die maximale Breite für Rockobjekte ist die Summe aus maximalem Beinabstand für Beinpaare zuzüglich der minimalen Breite für Beine (siehe Abb. 4.11).

In einem dritten Schritt wird eine Liste aller Objekte erstellt, die zuvor einer der drei Kategorien zuordnet wurden. Dabei werden alle Beine aus der Liste entfernt, die auch einem Beinpaar zugeordnet wurden. Zuletzt werden die Objekte nach ihrer Priorität sortiert. Diese sind so gewählt, dass die Wahrscheinlichkeit, dass es sich bei einem Objekt um eine Person handelt, maximiert wird. Aus diesem Grund haben Beinpaare die höchste Priorität, einzelne Beine eine mittlere Priorität und Rockobjekte eine niedrige Priorität.

Da Röcke im Normalfall so kurz sind, dass sie die Messungen des Laser-Scanners nicht beeinflussen, entstehen hierdurch keine negativen Einflüsse auf die Detektionsrate von weiblichen Personen. Wurden mehrere Objekte derselben Kategorie zugeordnet, so hat innerhalb dieser Gruppe das Objekt, das dem Roboter am nächsten liegt, die höchste Priorität.

Von den so ermittelten Positionen von Objekten, an denen Menschen vermutet werden, werden maximal die vier höchst Priorisierten mittels Gesichtsdetektion untersucht. Falls diese an keiner der Positionen ein positives Ergebnis liefert, dreht sich der Roboter um 120° und startet den Vorgang von vorn. Sollte auch dieses Mal keine Person erkannt werden, wiederholt er diesen Schritt ein weiteres Mal. Somit kann der Roboter Personen finden, die sich an einer beliebigen Position innerhalb des Radius von 5 Metern um ihn herum befinden, was mit einer 180°-Drehung nicht möglich wäre, da der Roboter Personen direkt neben sich nicht finden würde (siehe Abb. 4.12).

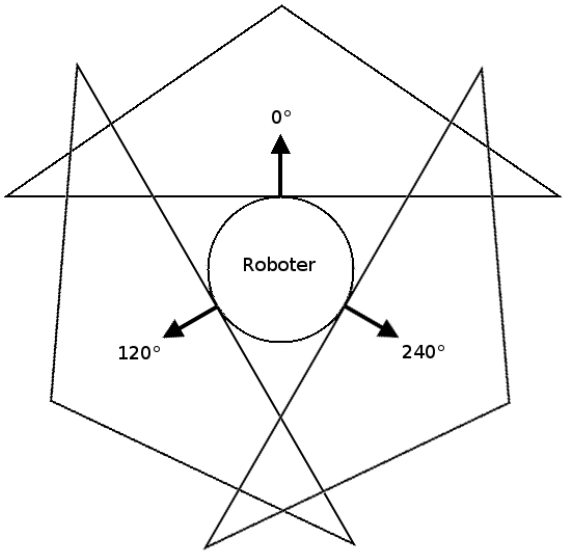


Abbildung 4.12: Drehungen des Roboters, um 360° zu erfassen

Die ermittelten Laserdaten werden im Administratorbereich der grafischen Oberfläche ausgegeben. Dabei werden nur die Objekte gefärbt, die eine der vier höchsten Prioritäten erhalten haben (siehe Abb. 4.13).

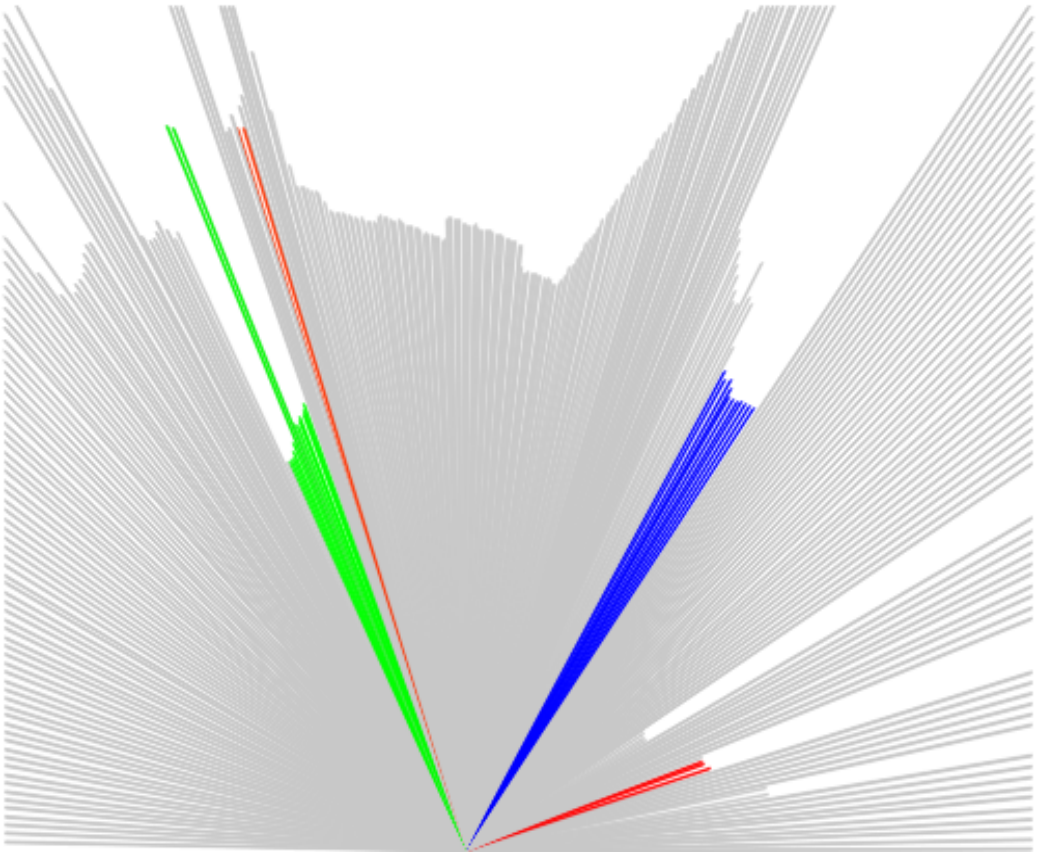


Abbildung 4.13: Prioritäten des Roboters. Beinpaar, Bein, Rockobjekt. Je heller desto niedriger die Priorität

4.5.1.3 Gesichtsdetektor

Zur Verifikation der ermittelten Positionen kommt ein Gesichtsdetektor zum Einsatz. Hier standen uns zwei Ansätze zur Verfügung, die dieses Problem schnell lösen: Der Viola-Jones- (VJ, siehe [20]) und der HoG-Detektor-Ansatz (HoG, siehe [22]). Wir haben uns für den HoG-Detektor entschieden, da der VJ nur auf der Frontalansicht von Gesichtern gut funktioniert und im Gegensatz zum HoG Gesichter in Profilansichten nicht detektiert.

Für die Gesichtsdetektion wird die bewegliche Kamera in die Richtung gedreht, in der zuvor eine mögliche Person durch den Beinpaardetektor ermittelt wurde. Hierbei kommt der beschriebene HoG-Detektor zum Einsatz, der sehr zuverlässig Gesichter erkennen kann. Dieser wird auf fünf aufeinanderfolgende Bilder angewendet. Falls in mindestens drei dieser Bilder, an ungefähr derselben Position in der Mitte des Bildes, ein Gesicht erkannt wird, wird dies als Bestätigung der Beinpaardetektormessung gewertet und die für diese Person ermittelten Koordinaten angesteuert. Zudem werden die Bilder im Administratorbereich inklusive Farbcodierung für die Anzahl der Detektionen je Pixelspalte der Bilder ausgegeben (siehe Abb. 4.14).

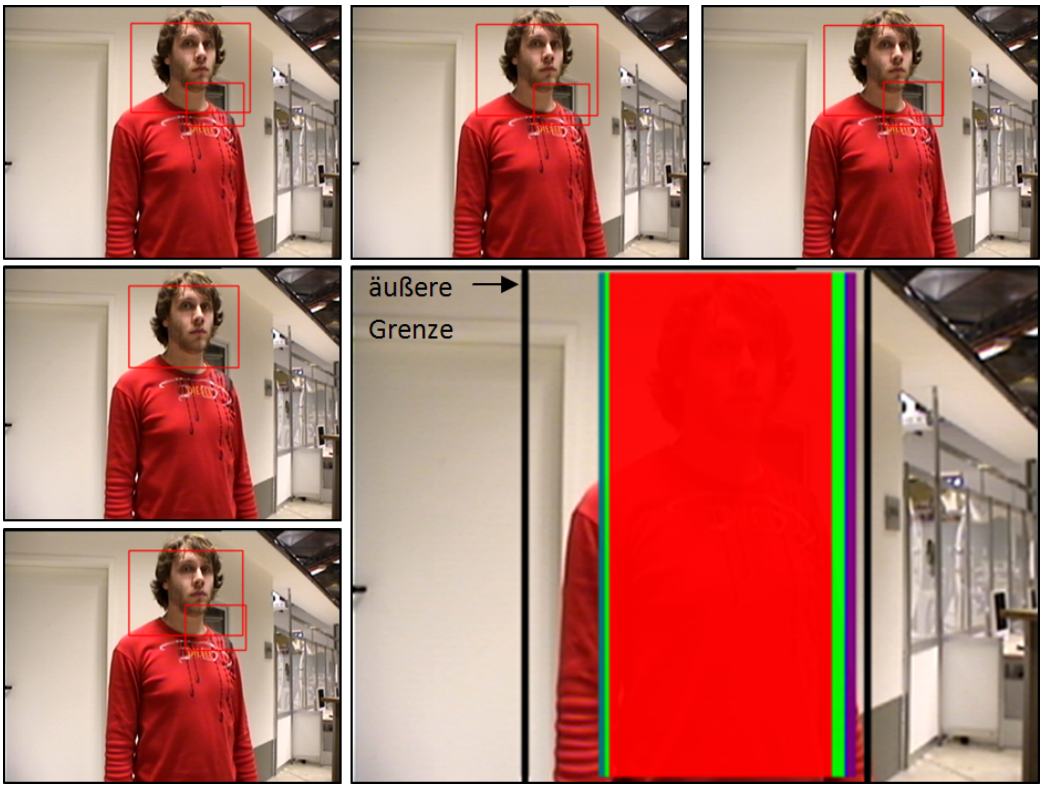


Abbildung 4.14: Ergebnis der Gesichtsdetektion. Anzahl Detektionen: Fünf. Vier. Drei. Zwei. Eins

4.5.2 Benutzerschnittstelle

Bisherige Tourguides nutzten für die Befehlseingabe durch den Nutzer vor allem fest installierte Knöpfe auf dem Roboter (siehe z.B. RHINO in [7]), eine Kombination aus Spracherkennung zusammen mit Knöpfen (siehe RoboX in [34]) oder ein Touchscreen und Knöpfe (siehe Jinny in [3]).

Für CampusGuide wurde ebenfalls eine Spracherkennung als Eingabemöglichkeit erwogen. Diese wurde jedoch schnell verworfen, da aufgrund des Einsatzgebietes (Außenbereich) sehr starke Nebengeräusche zu erwarten gewesen wären. Dieses hätte gerade bei längeren Sprachkommandos in einer hohen Fehlerquote resultiert. Zudem hätte die Art der Eingaben (vor allem Eigennamen der Mitarbeiter sowie Namen von Fakultäten, Gebäuden bzw. Abteilungen) den Aufbau einer sehr großen Erkennungsdatenbank erfordert, was speziell bei Eigennamen von Mitarbeitern zusätzlich dadurch erschwert worden wäre, dass man bei diesen oft sehr unterschiedliche Aussprachen erwarten kann (beispielsweise bei französischen, osteuropäischen oder asiatischen Namen).

Aus diesem Grunde haben wir uns entschieden den Touchscreen von CampusGuide als einzige Eingabemöglichkeit für den Nutzer zu verwenden und entwarfen ein Interaktionsszenario, in welchem der Touchscreen dem Nutzer eine intuitive Bedienoberfläche bietet. Der Benutzer wird von der Anforderung des Roboters bis zur Beendigung der Zielführung begleitet. Außerdem ermöglicht es dem Administrator, nach dem Einloggen auf erweiterte Systeminformationen zuzugreifen (siehe Abb. 4.15). Ebenso wurde in die meisten Bildschirmformulare die Möglichkeit eingebaut, die Emotionsausdrücke des Roboters zu sehen.

Nachfolgend sind die beiden Interaktionsszenarien mit der grafischen Benutzeroberfläche (GUI) näher beschrieben. Dies sind die Interaktion mit dem Benutzer und die Interaktion mit dem Administrator.

4.5.2.1 Interaktion mit einem Benutzer

Auf der Fahrt zum Aufenthaltsort des anfordernden Benutzers ist die grafische Benutzeroberfläche mittels eines pseudorandomisierten Passworts gesperrt und der Haupt-Emote-Bildschirm (siehe Abb. B.1) wird angezeigt.

Nach Ankunft des Roboters beim Nutzer kann sich dieser mittels Engabe des empfangenen Kennworts einloggen (Eingabeformular siehe Abb. B.2 und Abb. B.3) und aus einer der verfügbaren Zielauswahl-Listen (Fakultäten, Gebäude, Abteilungen und Personen) ein anzusteuerndes Ziel auswählen.

Zur vereinfachten Suche innerhalb des Zielauswahlformulars stehen dem Benutzer Filter zur Verfügung, mit deren Hilfe sich die Liste der Ziele ein-

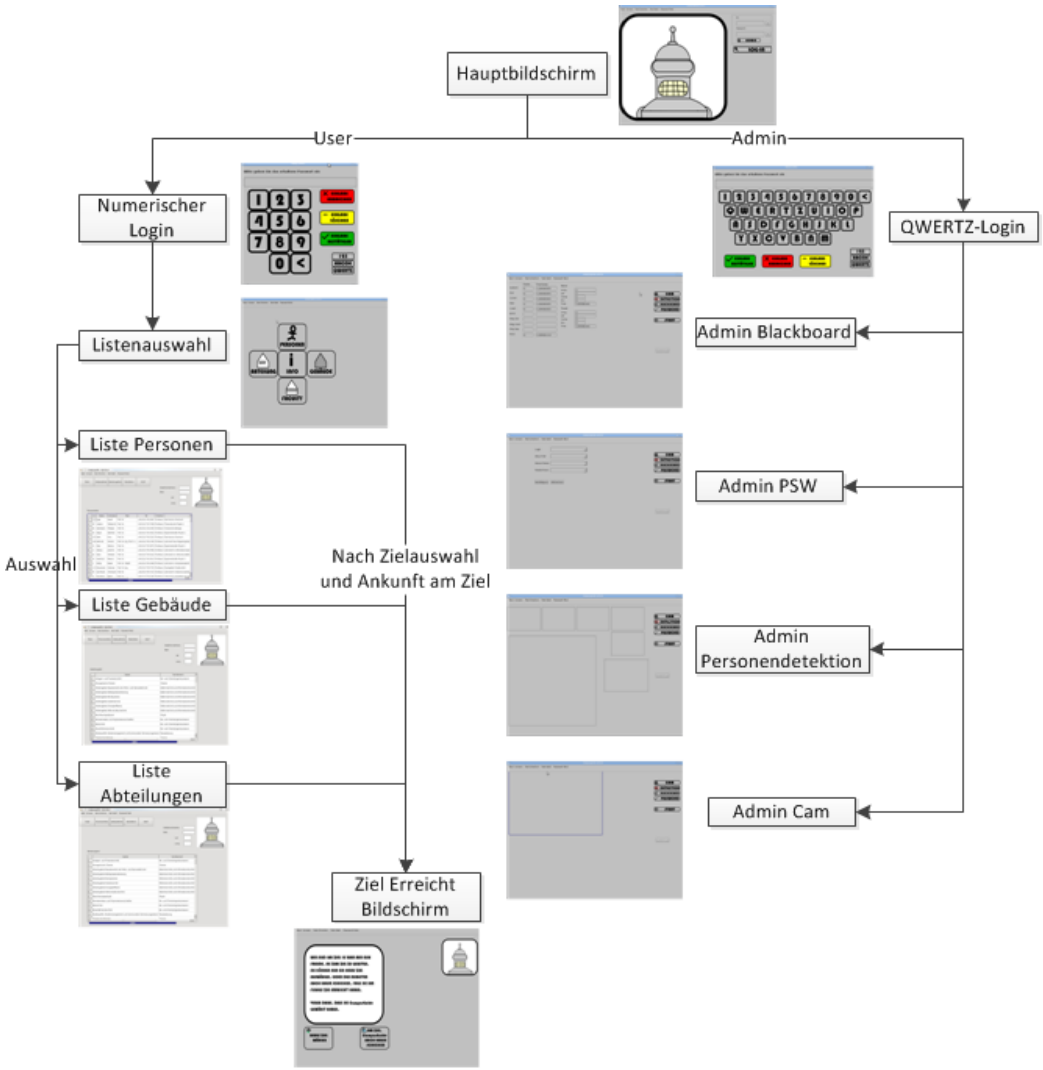


Abbildung 4.15: Interaktionsszenarien der GUI

schränken lässt.

Des Weiteren hat der Benutzer Zugriff auf ein Hilfemenü in dem die Grundlagen der Bedienung des CampusGuide sowie die Motivation des Projekts dargestellt sind (siehe Abb. B.8).

Nach Ankunft am Ziel hat der Nutzer im Ziel-Erreicht-Bildschirm die Möglichkeiten ein neues Ziel auszuwählen oder CampusGuide zu einer vordefinierten Heimatposition zu schicken (siehe Abb. B.7)

4.5.2.2 Interaktion mit einem Administrator

Eine Überwachung und Konfiguration des Systems ist im Administratorbereich möglich. Hier können die aktuellen Zustandsdaten des Roboters sowie Werte des Blackboards (siehe Abb. B.10) ausgelesen werden. Zusätzlich können Ergebnisse der Module Gesichts- und Personendetektion (siehe Abb. B.9 und B.12) visualisiert werden. Dieser Bereich ist mit einem änderbaren Passwort geschützt, was verhindert, dass dieser Bereich von Benutzern eingesehen werden kann (siehe Abb. B.11).

4.5.3 Emotionen

Um die Interaktion mit dem Benutzer zu erleichtern, wurde CampusGuide mit einem Emotions-Zustandsautomaten ausgestattet (siehe auch [35] zum Nutzen von Emotionen bei Robotern). Während andere Tourguides dazu mechanische Aufsätze benutzten, welche Mund und Augenbrauen darstellen (siehe Minerva in [2]), oder diese mittels einfacher Emoticons auf einer LED-Matrix darstellten (z.B. bei RoboX in [35]) wurde von uns für CampusGuide die GUI zur Emotionsdarstellung benutzt, wobei als Frontend ein selbsterstellter Avatar diente, welcher von uns auf Basis des Charakters Bender aus der Zeichentrickserie Futurama angefertigt wurde. [4]

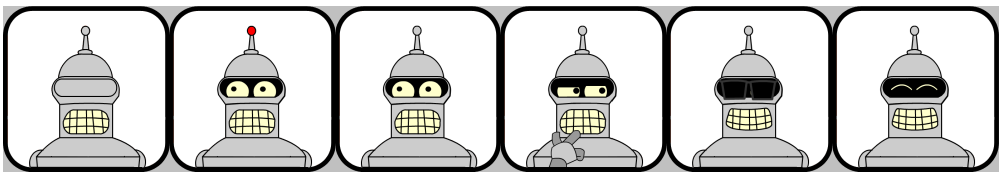


Abbildung 4.16: Emotionszustände 0-5 (v.l.n.r.)

[4]Bender and Futurama are copyright and trademark of FOX (http://www.comedycentral.com/shows/futurama/index.jhtml)

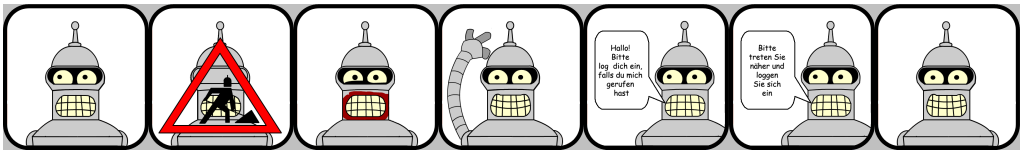


Abbildung 4.17: Emotionszustände 6-12 (v.l.n.r.)

Die Zustände des Emotions-Zustandsautomaten korrespondieren weitgehend mit den Zuständen des Gesamtsystems. Siehe Abbildung 4.18 der Zustandsnummern, Zustandsübergänge und korrespondierenden Bender-Emotionsausdrücke, welche in der nachfolgenden Tabelle 4.2 noch einmal tabellarisch aufgeführt sind.

Zustand	Bedeutung
0	Idle, wartend auf Benutzeranfrage
1	Beschäftigt (z.B. mit der Berechnung eines Weges zum anfordernden Benutzer)
2	Auf dem Weg zum anfordernden User
3	An ungefährer Userlocation angekommen, Suche nach User (Systemzustand = 3, GUI-Zustand = 1)
4	User hat Ziel gewählt, fahre hin
5	Roboter an Userlokation angekommen, warte auf Abschluss der Fahrt oder neue Zielwahl durch User
6	Roboter fährt zur Heimatposition
7	Fehlerzustand
8	Wartung
9	Habe eine Person in der Nähe entdeckt, Fahre hin
10	Zu potentiellem User gefahren, warte auf Einloggen des Users
11	Keinen potentiellen User gefanden, warte auf Einloggen des Users
12	User hat sich eingeloggt, warte auf Zielauswahl (entspricht Systemzustand = 3, GUI-Zustand = 5)

Tabelle 4.2: Übersicht über die Emotionszustände

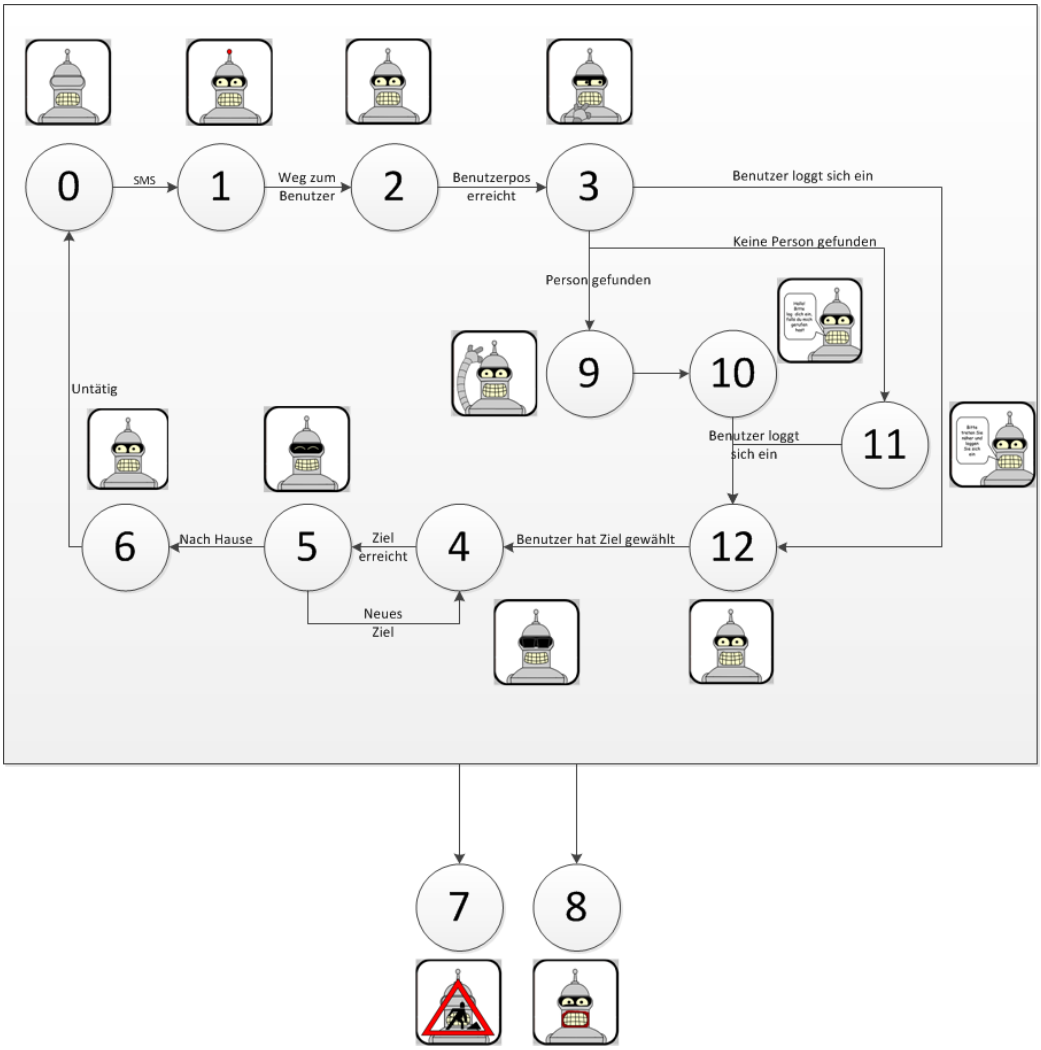


Abbildung 4.18: Übergänge der Emotionszustände

CampusGuide ist also am Anfang (Ruhezustand an Heimatposition) in Zustand 0 (Idle), geht bei Empfang einer Nachricht (und während der Kalkulation des Weges zum Benutzer) in Zustand 1 und schließlich, wenn er sich auf den Weg zum Benutzer macht, in Zustand 2, bis er schließlich an den GPS-Koordinaten des anfordernden Handys in Zustand 3 geht.

Danach startet er die Suche nach einer Person in unmittelbarer Nähe und geht dann entweder in Zustand 11 über (wenn er keinen Benutzer gefunden hat) oder fährt in Zustand 9 zu der entdedckten Person und geht dann in Zustand 10 über. Nach Einloggen des Users gelangt er in Zustand 12, nach Zielauswahl in Zustand 4, in dem CampusGuide dann zum Ziel fährt und beim Erreichen des Ziels in Zustand 5.

Nun kann der Roboter dann entweder wieder in Zustand 4 gelangen, oder in Zustand 6, je nachdem ob sich der Benutzer dazu entscheidet, sich zu einem neuen Ziel führen zu lassen, oder CampusGuide nach Hause schickt. Ist CampusGuide nach einer oder mehrerer Zielführungen wieder an seiner Heimatposition angekommen geht er in den Zustand 0, was den Zyklus vervollständigt.

4.6 Webserver

In diesem Abschnitt stellen wir das von uns implementierte Webinterface vor. Das Ziel dieser Anwendung im Kontext von CampusGuide ist die Überwachung der Position des mobilen Roboters. Der Grundgedanke dabei ist, eine Karte mit der Ortsanzeige des Roboters aufzubauen. Dazu wurde ein Client/Server-Modell implementiert. Auf dem Roboter läuft das Webinterface, welcher als Server interagiert. Dieses stellt die GPS-Koordinaten des Roboters dem Client zur Verfügung.

Diese Server/Client-Verbindung ermöglicht es dem Nutzer, jederzeit Informationen zur Position des Roboters abzufragen. Mittels des Clients wird hierbei eine TCP-Verbindung zum Webserver auf dem Roboter aufgebaut und periodisch die Position abgefragt. Die aktuelle Position des Roboters wird vom Smartphone im Blackboard abgelegt, vom Webserver ausgelesen und an den Client auf Anfrage geschickt.

Der Client benutzt zur Darstellung der Karte die API von OpenLayers, einer Schnittstelle zur Erzeugung dynamischer OSM-Karten mit JavaScript. In der grafischen Oberfläche wird eine Karte angezeigt, die zoombar und deren aktueller Bildausschnitt verschiebbar ist. Die Karte aktualisiert sich und den angezeigten Roboterpfad in zyklischen Abständen selbstständig. Der Pfad wird in einer externen GPS-Exchange-Datei gespeichert und aktualisiert, sobald eine neue GPS-Position beim Client angekommen ist. Bei jeder Aktualisierung der Karte wird ein neuer Wegpunkt auf der Karte sichtbar, so dass ein Pfad die gefahrene Strecke darstellt und ein zusätzlicher Marker die aktuelle Position des Roboters auf dem Pfad anzeigt. Ein Beispiel hierfür zeigt Abb. 4.19.

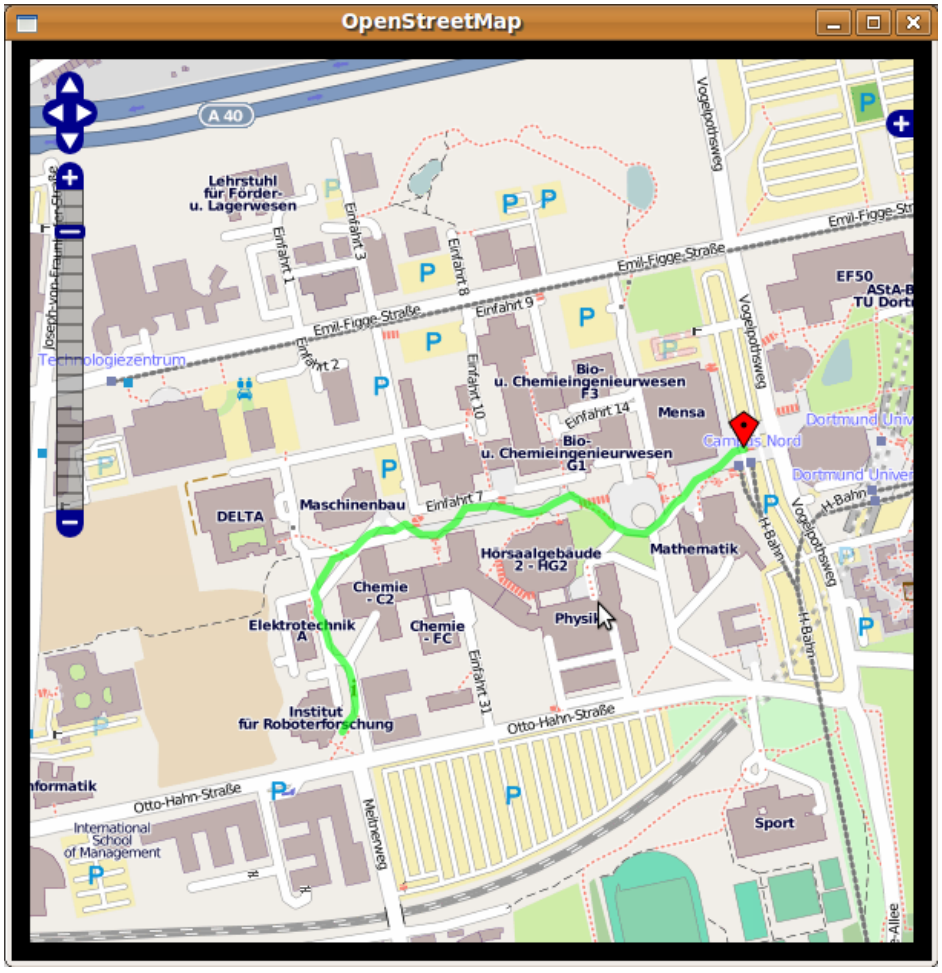


Abbildung 4.19: Karte der gefahrenen Strecke

Kapitel 5

Evaluierung

5.1 Bluetooth-Abstandsmessung

In Kapitel 4.2.5 wurde das Verfahren zur Abstandsmessung von CampusGuide zum Benutzer mittels *Bluetooth* vorgestellt. Dieses Verfahren soll nun quantitativ bewertet werden.

Um zu gewährleisten, dass die folgende Auswertung die tatsächliche Leistung des Verfahrens widerspiegelt, muss eine für das Szenario typische und repräsentative Menge an Daten aufgenommen werden. Da das Szenario in der Praxis so aussieht, dass ein Benutzer dem CampusGuide folgt und somit zumeist unmittelbar hinter oder vor dem Roboter läuft, werden die Testdaten auf ähnliche Weise aufgenommen. Es sei an dieser Stelle erwähnt, dass im Vorfeld Tests mit insgesamt sieben verschiedenen Handys im Hinblick auf die Bluetooth-Signalstärke durchgeführt wurden. Dabei konnten keine, für die Auswertung relevanten, Unterschiede in der Signalstärke festgestellt werden. Das Testszenario ist wie folgt aufgebaut. Die Messungen werden in einem, fünf und zehn Metern Abstand zum Roboter durchgeführt. Zu jedem dieser drei Abstände werden Messungen in den vier nachstehenden Kategorien aufgenommen:

- Benutzer hält das Handy in der Hand, dem Roboter zugewandt.
- Benutzer hält das Handy in der Hand, vom Roboter abgewandt.
- Benutzer hat das Handy in der Hosentasche, dem Roboter zugewandt.
- Benutzer hat das Handy in der Hosentasche, vom Roboter abgewandt.

Die Unterscheidung, ob der Nutzer dem Roboter zugewandt oder abgewandt ist, ist daher interessant, da es für das Szenario wichtig ist, ob sich der

Körper des Benutzers zwischen Handy und Roboter befindet oder nicht. Dies kann die empfangene Bluetooth-Signalstärke beeinflussen. Zur Evaluierung der Höhe in der das Telefon gehalten wird, werden Messungen angestellt, bei denen der Benutzer das Mobiltelefon in der Hosentasche trägt. Hierbei werden bessere Signalstärken vermutet, da sich die beiden Endgeräte, Smartphone des Roboters und das Benutzertelefon, auf gleicher Ebene befinden. Durch die Unterscheidungen beim Abstand und der Position des Handys ergeben sich insgesamt zwölf Testszenarien. Für alle Szenarien werden 100 Messungen vorgenommen. Zu beachten ist hierbei, dass die Zeit mit in die Messungen einfließt. Sollte nach einem Zeitraum von drei Sekunden kein neuer Signalwert gemessen worden sein, wird dies als Fehldetektion gewertet, was bedeutet, dass das Handy des Nutzers zu diesem Zeitpunkt nicht geortet werden kann. In diesem Fall wird ein Signalwert von -255 in die Ergebnisse aufgenommen. Dieser Wert stellt gleichzeitig das Minimum der Signalstärke dar. Das Optimum liegt bei einem Wert von 0.

Abbildung 5.1 zeigt beispielhaft das Szenario, in dem der Nutzer das zu ortende Handy in der Hand hält und dem Roboter zugewandt ist. Die Ergebnisse für die anderen Szenarien sind in Abschnitt B.2 als Kurvendiagramme dargestellt. Auf der Y-Achse ist die Signalstärke als blaue Linie für die 100 Messungen eingezeichnet. Betrachtet man alle Grafiken, erkennt man leicht, wie sich die einzelnen Szenarien voneinander unterscheiden. Die Notwendigkeit, die Zeit mit in unser System der Bluetooth-Abstandsmessung einfließen zu lassen, wird deutlich, wenn man die Signalstärke betrachtet. Diese bleibt bei allen betrachteten Szenarien auf einem gleichbleibenden Niveau. So ist es nicht möglich zu unterscheiden, ob sich der Nutzer in einem oder in zehn Metern Entfernung vom Roboter entfernt aufhält. Einzig die aus der verstrichenen Zeit gewonnenen Fehldetektionen geben einen Aufschluss darüber, wo sich das zu ortende Handy in etwa befindet. Betrachtet man die Ergebnisse der zusammengehörigen Kategorien für sich, also z.B. das das Handy vom Nutzer in der Hand gehalten wird und er dem Roboter zugewandt ist, so sieht man recht deutlich, dass die Zahl der Fehldetektionen mit zunehmendem Abstand steigt. Bei einem Abstand von zehn Metern wird in den meisten Messungen kein Signal empfangen. Ein weiterer wichtiger Punkt, der sich in den Ergebnissen widerspiegelt ist, dass es für die Ortung des Nutzerhandys besser ist, wenn dieser das Handy in der Hosentasche aufbewahrt. Das Handy ist dann in etwa auf gleicher Höhe mit dem Smartphone auf dem Roboter. Die Anzahl der Fehldetektionen ist in diesem Fall geringer. Das Niveau der empfangenen Signalstärke bleibt auch hier in beiden Fällen ununterscheidbar. Der letzte wichtige Punkt ist, ob der Nutzer dem Roboter zugewandt oder abgewandt steht, also der Körper des Nutzers zwischen seinem Handy und dem Roboter ist oder nicht. Auch hier zeigen die Ergebnisse recht deut-

lich eine Relevanz, was sich in der Anzahl der gestiegenen Fehldetektionen niederschlägt.

Für unsere letztendliche Realisierung der Abstandsmessung mittels *Bluetooth* müssen wir darauf achten, dass der Roboter nicht bei jeder Fehldetektion stehen bleibt, da er annimmt, der Nutzer sei nicht mehr in der Nähe. Da die Messungen selbst in einem Abstand von ca. einem Meter schon Fehldetektionen aufweisen, betrachten wir dabei die Menge der Fehldetektionen. Erst auf fünf aufeinander folgende Fehldetektionen reagiert unser System und nimmt an, dass sich der Nutzer mehr als zehn Meter vom CampusGuide entfernt hat. In diesem Fall wartet der Roboter darauf, dass der Benutzer sich wieder nähert. So werden fast alle Fehldetektionen, die sich bis zu einem Abstand von zehn Metern ereignen nicht falsch interpretiert. Erst ab einem Abstand von zehn Metern treten so häufig fünf Fehldetektionen nacheinander auf, dass unser System stehen bleibt und auf den Nutzer wartet.

Die Evaluierung zeigt, das dies ein guter Ansatz ist, um die Messungenauigkeiten der *Bluetooth*-Technologie zu kompensieren.

5.2 Befahrbarkeit

Um zu entscheiden, mit welchen Parametern wir den kamerabasierten Ansatz der Befahrbarkeitserkennung betreiben, haben wir das System nach der Realisierung evaluiert. Hierzu berechnen wir verschiedene Kombinationen aus den in Kap. 4.4 vorgestellten Merkmalen mit jeweils unterschiedlichen Parametern und trainieren damit Netze unterschiedlicher Topologien. Diese Netze werden im nachhinein getestet und ausgewertet, um die Klassifikationsergebnisse zu vergleichen.

5.2.1 Datenaufnahme

ő

Der eigentlichen Evaluierung vorausgehend steht die Datenaufnahme, in der die Bilder aufgenommen werden, welche später für das Training, die Validierung und den Test verwendet werden. Die Menge der Daten sollte möglichst groß und repräsentativ sein, um bestmögliche Ergebnisse zu liefern. Da der Außeneinsatz von CampusGuide mit der aktuellen Roboterhardware nicht möglich ist, haben wir entschieden, nur verhältnismäßig wenig Daten an unterschiedlichen Orten aufzunehmen. Es wurden lediglich die Zugangswege zum Institut für Roboterforschung (IRF) in einem Umkreis von ca. 15 m zur Datenaufnahme benutzt. Im Gegensatz dazu haben wir vermehrt die unterschiedlichen Beleuchtungssituationen bei verschiedenen Tageszeiten

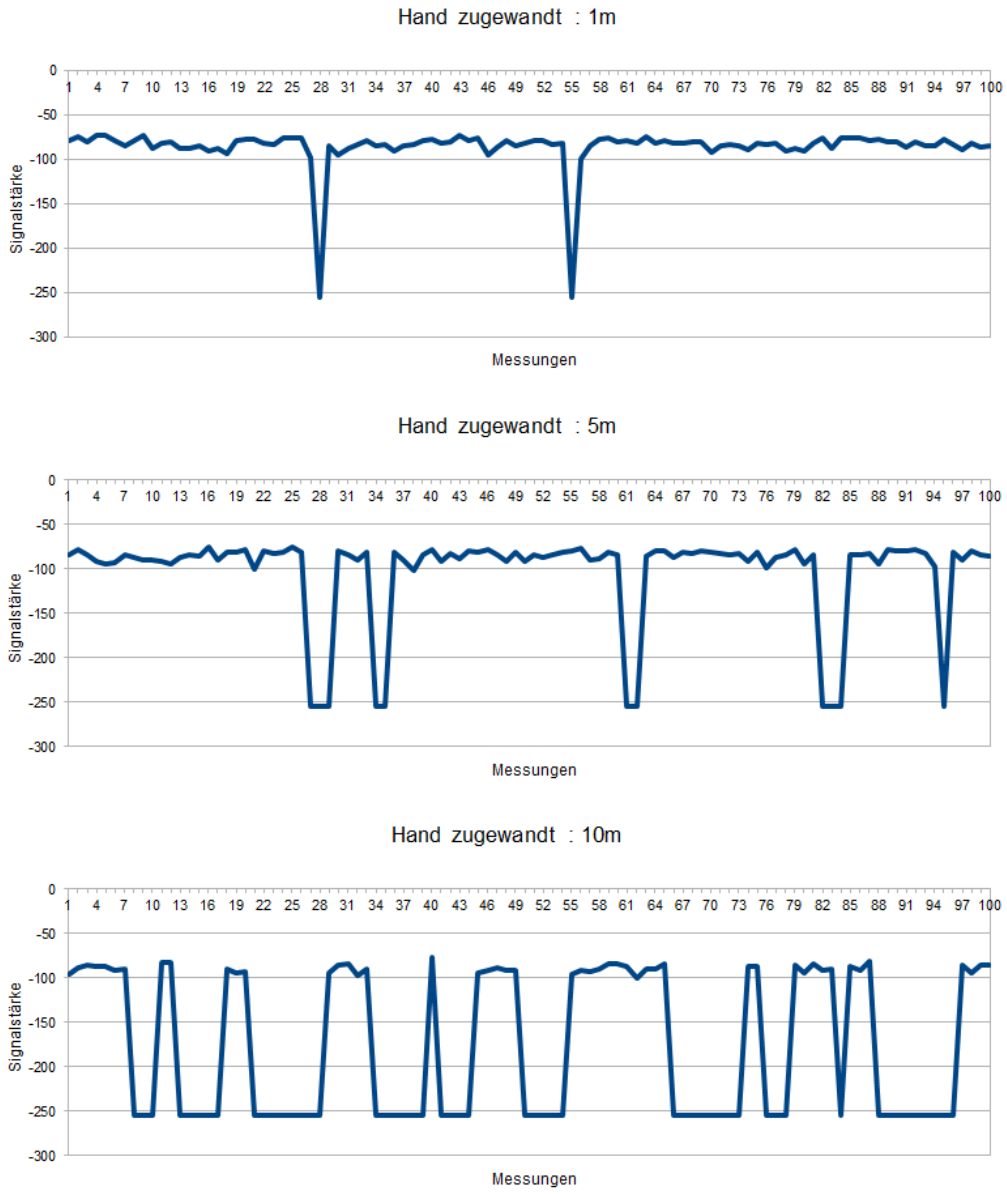


Abbildung 5.1: Ergebnisse von 100 Messungen. Der Nutzer hat hierbei das Handy in der Hand und ist dem Roboter stets zugewandt.

und Wetterbedingungen berücksichtigt, um so ein möglichst repräsentatives Ergebnis zu erhalten. Zu diesem Zweck haben wir an vier Tagen, zu jeweils unterschiedlichen Uhrzeiten, bei variablen Wetterbedingungen, Bilder aufgenommen. Diese 571 Bilder entstanden indem der Roboter langsam geschoben wurde und die Bilder der Kamera im Sekundenabstand gespeichert wurden. Im nächsten Schritt folgte die Annotation der Bilder mit dem in Kap. 4.4.3 vorgestellten Programm. Hier wurde jedoch die Granularität der Codierung gröber gewählt, um nur noch die Möglichkeiten befahrbar und nicht befahrbar zu haben.

Nachfolgend wurden die Daten disjunkt in Trainings-, Validierungs- und Testmenge im Verhältnis $4 : 1 : 1$ aufgeteilt.

5.2.2 Auswahl der Merkmale

Wie schon in Kap. 4.4.4 beschrieben, haben wir jedes Bild mittels *sliding windows* in 165 einzelne Fenster aufgeteilt. Diese Fenster überlappen sich jeweils um die Hälfte auf beiden Achsen. Aus jedem Fenster werden Merkmale extrahiert, wobei es für diese Merkmale mehrere Auswahlmöglichkeiten gibt. Letztendlich haben wir folgende Merkmalsvektoren evaluiert:

- Farbhistogramm
- Histogramm der Gaborfilterantworten
- Farbhistogramm und Histogramm der Gaborfilterantworten

Bei dem Farbhistogramm-Merkmal besteht der Merkmalsvektor aus einem 10×10 Histogramm, also dementsprechend aus 100 normalisierten Histogrammwerten.

Das Histogramm der Gaborfilterantworten besteht aus 8×4 Werten, definiert durch die Anzahl an Orientierungen und Frequenzen. Jedes *bin* beinhaltet die, in der Gesamtheit normalisierte, Summe aller Pixelwerte in der Antwort des Filters. Je mehr Struktur im Bild erkannt wird, desto näher ist der Wert im *bin* an 1,0.

Die Kombination der beiden Varianten ist eine Konkatenierung der Merkmalsvektoren und ist dementsprechend hier 132 Werte groß. Durch die vorherige Normierung beider Vektoren ist der Definitionsbereich identisch und kann das Ergebnis nicht verfälschen.

Jede dieser drei Varianten wird zusätzlich mit mindestens zwei Parametersätzen evaluiert. Wie in Kap. 4.4.4.1 beschrieben, nutzen wir drei verschiedene helligkeitsunabhängige Farbräume (Parameter COLSPACE), bei denen jeweils die Farbkanäle für die Histogramme gewählt werden und der Helligkeitskanal vernachlässigt wird. Die Histogramme der Gaborfilterantworten

werden mit verschiedenen Parametern für die Gaborfilter erstellt. Hierbei haben wir jedoch für die meisten Parameter die Standard-Werte der verwendeten *Gaborjets* benutzt. Die Orientierungen sind mit acht Variationen gut gewählt, denn so können Kanten mit vertikaler und horizontaler Ausrichtung sowie zusätzliche Kanten, die $22,5°$ von diesen Achsen gedreht sind, erkannt werden. Zudem haben wir lediglich vier verschiedene Frequenzen gewählt, um die Datenmenge nicht zu groß und die Berechnungen damit zu aufwändig für eine Echtzeitklassifikation zu machen. Als Frequenzparameter wurden Minimum (MINF) / Maximum (MAXF) auf die Standard-Parameter 0,125 / 1,0 sowie die selbstgewählten Parameter 0,4 / 0,5 gesetzt. Das erste Wertpaar beschreibt einen sehr großen Frequenzbereich, was zur Folge hat, dass auch die meisten Frequenzen aus dem Bild verwendet werden. Informelle Tests führten zum zweiten Parametersatz.

5.2.3 Training der neuronalen Netze

Dadurch, dass die Größe der Merkmalsvektoren variiert, müssen auch die Netze dementsprechend anders aufgebaut werden. Hier wurden mehrere Varianten evaluiert. Zum einen die Anzahl der Eingabeneuronen, je nach Merkmalsvektor, und zum anderen die Anzahl der Neuronen in der versteckten Schicht. Letztere wurde in Schritten von fünf Neuronen bis zum Maximum von 100 Neuronen in der versteckten Schicht evaluiert (Parameter NEUR). Dieses Maximum wird bestimmt durch die Anzahl der Trainingsdaten im Verhältnis zu den freien Parametern, so dass die Trainingsdatensätze die Anzahl der freien Parameter nicht um einen Faktor 10 überschreiten. Somit ergibt sich eine Gesamtanzahl von 220 evaluierten neuronalen Netzen, welche alle nur ein Ausgabeneuron aufgrund der binären Entscheidung zwischen befahrbar und nicht befahrbar haben. Beim Training dieser Netze wurde mittels der Validierungsdaten ein Auswendiglernen der Trainingsdaten (*overfitting*) überprüft und dieser Effekt minimiert.

Abschließend werden die neuronalen Netze zur Evaluierung getestet und die Ausgaben mit den Annotationen der Testdaten verglichen. Um auch hier den optimalen Schwellenwert für das Ausgabeneuron des Netzes zu finden, verwenden wir *receiver operating characteristic*-Kurven (ROC-Kurven), die die relativen richtigen Detektionen (*true positives*) mit den Detektionen vergleicht, die eigentlich nicht befahrbar sein sollten, jedoch vom Netz als befahrbar markiert worden sind (*false positives*). Relativ bedeutet hierbei, dass die Anzahl der *true positives* ins Verhältnis zur Anzahl der als befahrbar markierten Bereiche in der Annotation gesetzt werden, und dass die Anzahl der *false positives* dementsprechend ins Verhältnis mit der Anzahl der, als nicht-befahrbar markierten Bereiche gesetzt werden. Durch die Variati-

on des Schwellenwerts in 100 linearen Schritten von 0,0 bis 1,0 ergibt sich die vorhin erwähnte ROC-Kurve. Auf dieser Kurve befindet sich ein für uns wichtiger Punkt, an welchem sich die *equal-error-rate* (EER) ablesen lässt. Dieser Wert wird erreicht, wenn die Anzahl der *false positives* und der *false negatives* gleich ist. Hier ist demnach das Verhältnis zwischen *false positives* und *true positives* optimal. Nachfolgend bewerten wir die Ergebnisse.

5.2.4 Ergebnisse

Durch die Vielzahl der Netze ist es unmöglich alle Ergebnisse an dieser Stelle exakt zu analysieren, daher beschränken wir uns hier auf die besten drei Netze.

MODE	COLSPACE	MINF	MAXF	NEUR	Detektionsrate
0	NRG	0,125	1,0	60	88,625%
1	HSV	-	-	30	86,685 %
2	-	0,125	1,0	90	79,237 %

Tabelle 5.1: Parameter der besten Netze (- bedeutet nicht benötigt)

Bei diesen Netzen (siehe Tab. 5.1)handelt es sich jeweils um das beste Netz der verschiedenen Merkmalsvektoren, Farbe (MODE1), Textur (MODE2) und Farbe+Textur (MODE0). Andere Netze des gleichen Modus haben schlechtere Detektionsraten. Der Gaborfilter als Merkmal liefert die schlechtesten Ergebnisse mit Raten von 47,02% bis 79,237%. In der Realität erweisen sich diese Ergebnisse als noch schlechter als die reinen Detektionsraten schließen lassen.

Die Detektionsergebnisse (siehe Abb. 5.3) erscheinen willkürlich und liefern durch die vielen Fehldetektionen am unteren Bildrand kein brauchbares Ergebnis für die Verwendung mit dem Laserscanner. CampusGuide würde sehr häufig stehenbleiben oder fälschlicherweise Hindernisse erkennen. Dies spiegelt sich auch in der ROC-Kurve wieder (siehe Abb. 5.2). Aus dem Graph lässt sich ablesen, dass mit kleiner werdendem Schwellenwert sowohl Detektionsrate als auch Falschdetektionsrate relativ schnell wachsen. Der optimale Schwellenwert liegt hier bei 0,68, da hier das Verhältnis zwischen nicht erkannten befahrbaren Bereichen und fälschlicherweise als befahrbar markierten nicht befahrbaren Bereichen nahe 1 : 1 ist. Dies liegt vermutlich daran, dass der Abstand der Fugen durch die Perspektive verzerrt wird und so die Fugen am unteren Bildrand größer scheinen, als die am oberen Bildrand. Die gleiche Verzerrung kann man bei nicht befahrbaren Untergründen feststellen, denn auch hier wird die meist sehr feine Struktur im oberen Bildbereich zu

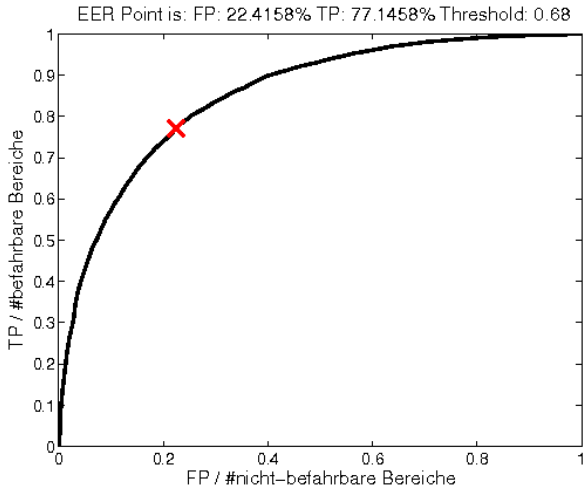


Abbildung 5.2: ROC-Kurve des besten MODE2-Netzes

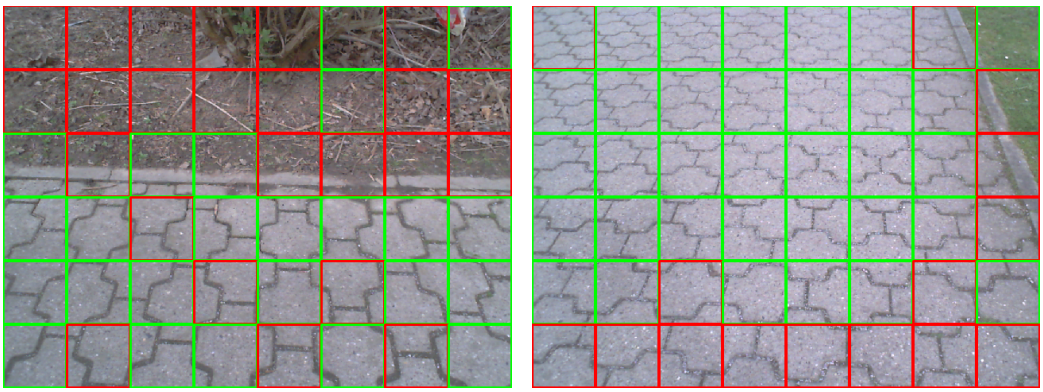


Abbildung 5.3: Klassifikationsergebnisse der Gaborfilter. (Grün = befahrbar, Rot = nicht befahrbar)

einem großen Teil unkenntlich. Diesen Verzerrungen könnte man entgegenwirken, indem sich die Frequenz der Filter je Bildbereich automatisch den Abständen der Fugen anpasst, sodass die Frequenzen im unteren und oberen Bildbereich zu den Fugen äquivalent sind. Ein weiteres Problem dieses Merkmals ist der enorme Rechenaufwand, der aufgrund der Faltung der Filter mit dem Bildausschnitt entsteht. Im Durchschnitt braucht die Klassifikation ca. 2600 ms, was bei diesen schlechten Detektionsergebnissen deutlich zu lang ist.

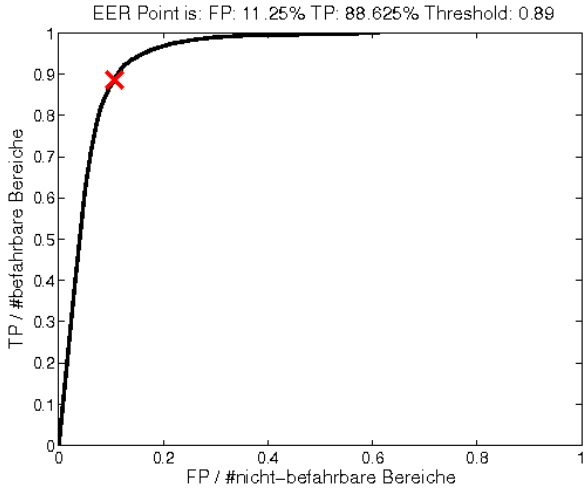


Abbildung 5.4: ROC-Kurve des besten MODE0-Netzes

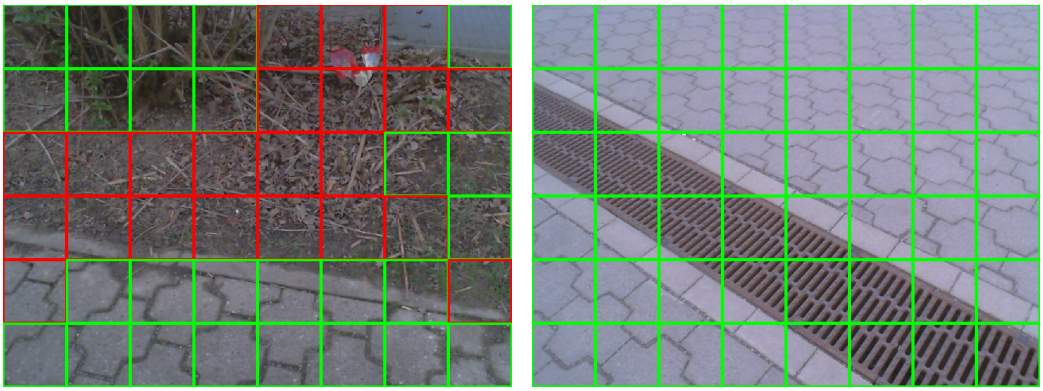


Abbildung 5.5: Klassifikationsergebnisse der Farbhistogramme und der Gaborfilter. (Grün = befahrbar, Rot = nicht befahrbar)

Bei der Nutzung beider Verfahren als Merkmale wurden die besten Ergebnisse erzielt (siehe Tab. 5.1). Die Klassifikationen (siehe Abb. 5.5) sehen wesentlich besser aus und der Klassifikator hat bis auf vereinzelte Fehldetektionen den Untergrund richtig erkannt. Hier ist besonders hervorzuheben, dass dieser Klassifikator, im Gegensatz zu den Anderen, auch für befahrbaren Boden untypische Strukturen und Farben erkennt wie z. B. Regenablaufrinnen. Im Gegensatz zu dem MODE2-Netz zeigt die ROC-Kurve für die kombinierten Merkmale (siehe Abb. 5.4) eine andere Ausprägung. Hier steigt erst die Anzahl der *true positives* sehr schnell, wobei die *false positives* nur langsam steigen. Ab einem bestimmten Punkt nahe des EER-Punkts invertiert sich dieses Verhältnis jedoch und die Falschdetektionsrate nimmt schnell zu. Der optimale Schwellenwert liegt bei 0,89. Diese Merkmalskombination ist wesentlich stabiler gegenüber der Bildverzerrungen durch die Kamera, hat jedoch das gleiche Geschwindigkeitsproblem. Die Klassifikation eines Bildes dauert hier durchschnittlich 2700 ms. Wenn man im Vergleich dazu die maximale Geschwindigkeit des Roboters von 1,4 m/s nimmt, würde dies bedeuten, dass der Roboter zwischen den Klassifikationen eine Strecke von mehr als 3,7 m zurücklegen könnte. Dies würde zur Folge haben, dass der Roboter, aufgrund seiner maximalen Sichtweite von 1,89 m, in eine unklassifizierte Umgebung führe. Somit ist auch diese Merkmalskombination in der Realität ohne Optimierungsaufwand nicht einsetzbar.

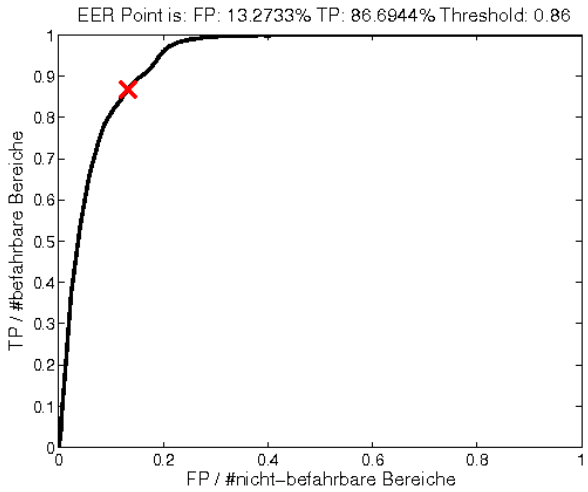


Abbildung 5.6: ROC-Kurve des besten MODE1-Netzes

Die letzte Variante, die ausschließlich die Farbe des Untergrunds berücksichtigt (siehe Abb. 5.7), erzielte laut der Evaluierung die zweitbesten Ergeb-

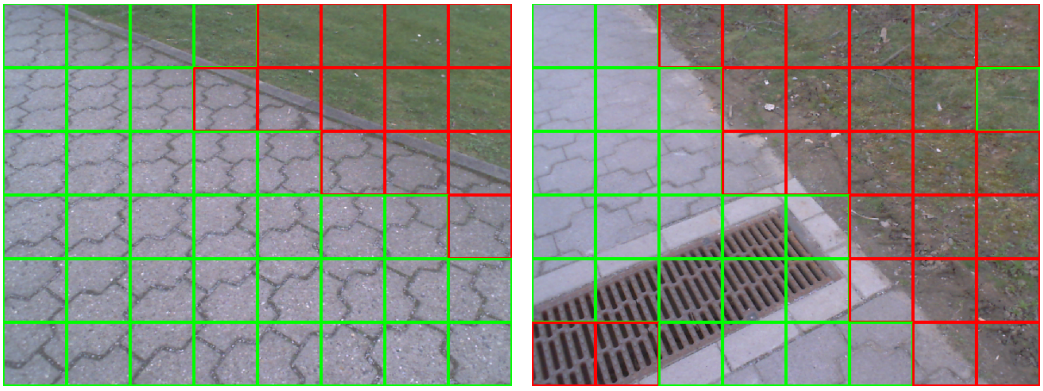


Abbildung 5.7: Klassifikationsergebnisse der Farbhistogramme. (Grün = befahrbar, Rot = nicht befahrbar)

nisse. Das einzige Problem bei diesen Merkmalen ist, dass die schon angesprochenen untypischen Bodenfärbungen durch Gitter, Roste oder nasse Stellen als nicht befahrbar markiert werden. Jedoch lässt sich dieses Problem auch im positiven Sinne nutzen, da der Roboter so auch nicht in Pfützen fahren kann. Die Besonderheit dieser Merkmale ist die Geschwindigkeit. Ein Bild lässt sich im Durchschnitt in 45 ms klassifizieren und ist somit schnell genug, um dem Roboter im Außeneinsatz frühzeitig Klassifikationsergebnisse zur Verfügung zu stellen.

Die ROC-Kurve für dieses Netz (siehe Abb. 5.6), verläuft zwar nicht so gut wie die des MODE0-Netzes, jedoch ist eine Detektionsrate von fast 87% ausreichend für die Verwendung mit dem Laserscanner.

5.3 Personendetektion

Wenn der Roboter an der Position angekommen ist, zu der er vom Benutzer gerufen wurde, soll er diese Person finden und sich auf sie zu bewegen. Hierzu verwenden wir zwei Werkzeuge. Das eine ist ein Beinpaardetektor, welcher mithilfe des Lasersensors funktioniert und etwa 40cm über dem Boden versucht Personen zu finden, indem er Objekte als Beine identifiziert. Eine so getroffene Vorauswahl wird danach durch das zweite Werkzeug, den Gesichtsdetektor, verifiziert. Hierzu wird die Kamera in die zuvor ermittelten Richtungen gedreht und an diesen Positionen nach Gesichtern gesucht. Sobald solch eine Verifikation erfolgt ist, fährt der Roboter zur gefundenen Person. Um dem Nutzer lange Wartezeiten zu ersparen, ist es wichtig, dass diese Detektion so schnell wie möglich durchführbar ist.

5.3.1 Beinpaardetektor

In diesem Abschnitt wird der Beindetektor evaluiert. Dabei soll herausgestellt werden, wie oft der Detektor Personen als solche identifiziert und von Objekten unterscheidet.

Vorbereitung Da die Auswertung durch die Kamera sehr zeitintensiv ist, ist es notwendig eine Vorauswahl zu treffen, und mögliche Ziele zu ermitteln, die dann von der Kamera aufgenommen und vom Gesichtsdetektor untersucht werden.

Der Laser-Scanner liefert für jeden Laserpunkt zwei Informationen, den Winkel, in dem er misst, und die Distanz, in der er auf ein Objekt trifft. Dadurch bietet sich dem Beinpaardetektor eine Möglichkeit, sehr schnell Daten auszuwerten, da die durch den Lasersensor aufgenommenen Daten in Echtzeit verarbeitbar sind. Aus diesem Grund bietet es sich an, im Zuge der Evaluation möglichst optimale Parameter zu finden, um eine möglichst hohe Wahrscheinlichkeit zu haben, dass der Gesichtsdetektor nur Positionen untersucht, an denen sich tatsächlich eine Person befindet.

Der Fokus liegt hierbei auf der Identifikation von Beinpaaren, um möglichst sicherzustellen, dass Personen, wenn sie im Sichtbereich des Lasers sind, auch gefunden werden. Falsch detektierte Objekte werden dann durch den Gesichtsdetektor abgewiesen.

Als Testdaten wurden insgesamt 200 Laserdatensätze aufgenommen (Datensatz ALL), von denen die Hälfte ein Beinpaar enthält (Datensatz PAIR), das vom Algorithmus detektiert werden soll. Die andere Hälfte enthält sowohl Daten ohne Person im Sichtbereich, als auch Personen, deren Beine durch einen Rock oder ähnliches verdeckt sind, sowie verdeckte Personen, von denen nur ein Bein sichtbar ist, und Personen deren Beine für den Algorithmus nur als ein Objekt erkannt werden. Es wurde speziell darauf geachtet, dass die verwendeten Daten möglichst an verschiedenen Positionen aufgenommen wurden, um möglichst große Variabilität in die Testdaten zu bringen.

Zur Verarbeitung wurden die Testlaserdatensätze jeweils in eine Datei geschrieben mit Informationen für jeden Laserpunkt bzgl. Winkel, Distanz zum Roboter und Zugehörigkeit zur Person. Dabei wird letztere binär kodiert, ein Laserpunkt gehört also entweder zu einem Bein, oder nicht. Insbesondere ist zu beachten, dass nicht getestet wird, ob ein Objekt gefunden wird, sondern ob es auf die höchste Priorität zur Gesichtsdetektion gesetzt wird. Wenn ein Rockobjekt zwar gefunden wird, jedoch an die zweite Position der Prioritätsliste gesetzt wird, weil fälschlicherweise ein Bein detektiert wird, so wird dies als nicht gefunden interpretiert.

Zur Bestimmung der Güte eines Parametersatzes wird jeder Laserdaten-

satz auf einer Skala von 0 bis 1 bewertet, wobei 1 für komplett richtig und 0 für komplett falsch steht. Komplett richtig heißt hier, dass sämtliche annotierten Punkte eines Beines auch erkannt wurden, bzw. der Algorithmus kein Objekt findet, wenn kein solches annotiert wurde. Die Güte ergibt sich dann aus dem Mittelwert aller Laserdatensätze des verwendeten Testdatensatz. Das Vorgehen ist dabei wie folgt:

1. Wurden Laserpunkte mit 1 annotiert (ANO)?,
2. Wurde ein Objekt vom Programm mit dem Label 5 (also ausgewählt) versehen (DET)?
3. Gibt es einen Laserdatenpunkt, der sowohl 1 als auch 2 erfüllt (TRUE)?

Dies wird in Tabelle 5.2 genauer aufgeschlüsselt, wobei RESULT die Güte beschreibt. Wenn alle drei Kriterien erfüllt sind ergibt sich die Güte des Parametersatzes aus dem Anteil der korrekt gefundenen Punkte von den annotierten Punkten.

ANO	DET	TRUE	RESULT	Grund
n	n	n	1	Kein Objekt in Datensatz. Korrekt erkannt.
n	n	j	0	-
n	j	n	0	Falsch detektiert.
n	j	j	0	-
j	n	n	0	Nicht gefunden.
j	n	j	0	-
j	j	n	0	-
j	j	j	#found/#leg	Korrekt erkannt. Anteil gefundener Beine.

Tabelle 5.2: Wertung für Laserdatensatz. n = Kriterium trifft nicht zu. j = Kriterium trifft zu. - = Fehler, kann nicht auftreten.

Testläufe Für die Testläufe wurden insgesamt sieben Parameter getestet, die einen Einfluss auf die Güte des Beinpaardetektors haben. Fünf Parameter beziehen sich dabei auf das Bein selbst: Minimale Breite (MINW), maximale Breite (MAXW), minimale Tiefe (MIND), maximale Tiefe (MAXD) und die Form (FORM). Zudem wurde der Abstand zwischen den Beinen bewertet (LEGDIST), sowie der Wert, der den Entfernungsunterschied zwischen zwei Messpunkten bestimmt, sodass an dieser Stelle ein neues Objekt gekennzeichnet wird (NEWOBJ).

Um Startparameter zu finden, wurde für einen einzelnen Laserdatensatz ermittelt, welche Parameter ausreichend waren, um das Beinpaar, das im

Sichtbereich war, zu detektieren. Hierbei ergaben sich die Werte in Tabelle 5.3.a START und eine Güte von 73%.

Nächstes Ziel war eine Annäherung an die optimalen Werte für die einzelnen Parameter. Zu diesem Zweck wurden sämtliche Parameter auf fünf Stufen getestet, zwischen denen jeweils für jeden Parameter ein vordefinierter Abstand lag, und deren mittlere Stufe dem Wert aus START entsprach (siehe Tab. 5.3.b IMPRO). Somit konnte eine maximale Güte von 87% erreicht werden.

Zur Bestimmung der final verwendeten Parameter wurde jeweils genau ein Parameter aus IMPRO verändert, während die anderen Parameter unverändert blieben. Dabei wurden sehr kleine Schrittweiten und sehr viele Schritte verwendet, um möglichst optimale Parameter zu ermitteln. Mit diesem Verfahren konvergierten die Parameter gegen die Werte in Tabelle 5.3.c FINAL und eine Güte von 93%.

Name	NEWOBJ	LEGDIST	MINW	MAXW	MIND	MAXD	FORM	Güte
a START	0.5	1.0	0.1	0.4	0.08	3	0.375	73%
b IMPRO	0.3	0.8	0.1	0.3	0.03	2.0	0.35	87%
c FINAL	0.23	0.48	0.115	0.32	0.04	1.85	0.35	93%

Tabelle 5.3: Iterative Verbesserung der Parameter

Auswertung Für die so ermittelten Werte ist mehr oder weniger eindeutig, welcher Wert das optimale Ergebnis liefert.

Für den minimalen Abstandsunterschied zwischen zwei Punkten zur Separation von Objekten (siehe Tab. B.16) wird offentsichlich, dass ein bestimmter Abstand zwischen zwei Punkten notwendig ist, um einzelne Objekte zu extrahieren. Ein Abstand von 5 cm reicht aus, um 70% der Objekte zu detektieren. Zwischen 22 und 23 cm wird so der Optimalwert von 93% erreicht. Höhere Werte liefern schlechtere Ergebnisse, weil hierdurch Beine eventuell mit anderen Objekten verschmelzen, wenn die Person nah an diesen steht. Bei niedrigeren Werten hingegen besteht das Problem, dass einzelne Punkte eventuell nicht zum selben Bein gehörend erkannt werden und somit das Bein nicht detektiert werden kann. Aus Tabelle B.16 wird zudem ersichtlich, dass der Verlauf für alle Testdatensätze ähnlich verläuft, wie der Beinpaar-Datensatz. Dies liegt hauptsächlich daran, dass Menschen in seitlichen Positionen, bei denen beide Beine versetzt sind, wie ein Objekt erscheinen und somit nicht eindeutig für den Beinpaardetektor erkennbar sind. Sie werden dann nur in relativ störungsfreien Umgebungen gefunden.

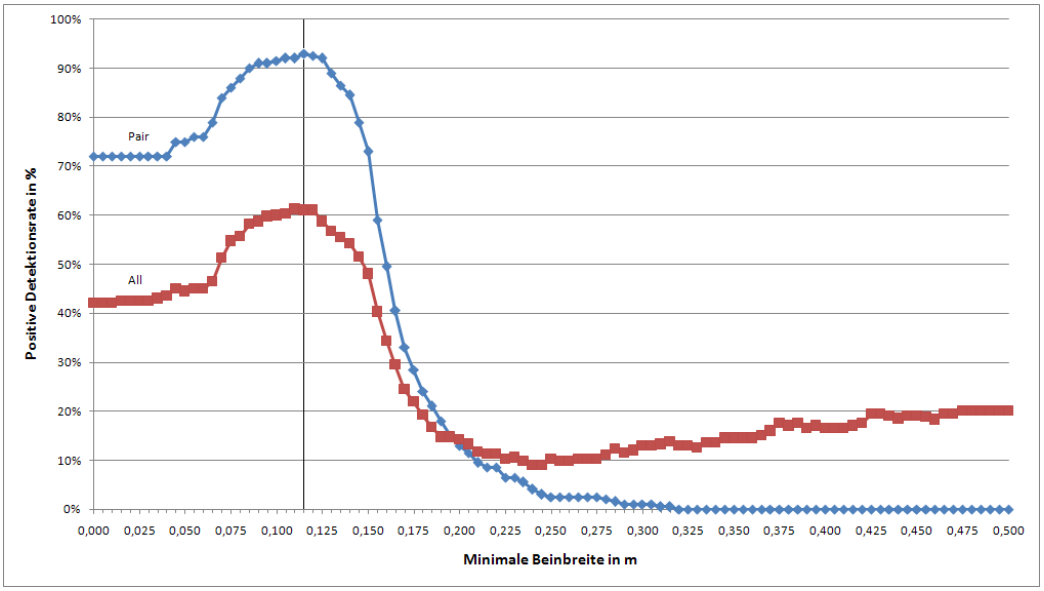


Abbildung 5.8: Verlauf bei Veränderung des Parameters MINW. $\diamond$ = Datensatz PAIR. $\square$ = Datensatz ALL

Das wichtigste Kriterium für das Erkennen eines Objektes als Bein ist seine Breite. Insbesondere die minimale Beinbreite hat einen starken Einfluss darauf, ob ein Objekt als Bein klassifiziert wird (siehe Abb. 5.8). Ist der Wert zu niedrig, so werden auch andere Objekte, wie z.B. Stuhlbeine oder Pfosten, als Beine erkannt. Da die Werte ausschließlich auf bekleideten Beinen optimiert wurden ist der Wert von 11,5 cm ein wenig höher als die tatsächliche Breite eines Beines, genügt aber um 93% der Beinpaare zu detektieren. Je höher der Wert jedoch im Vergleich zur Breite des Beines ist, desto unwahrscheinlicher wird das Bein als solches erkannt. Mit dem Datensatz ALL fällt auf, dass der Verlauf bis etwa 20 cm ähnlich ist wie bei PAIR, danach jedoch eine bessere Detektionsrate liefert und danach weiter steigt. Zu erklären ist dies damit, dass mit dem Steigen der minimalen Beinbreite weniger Objekte als Bein oder Rockobjekt, da diese Kategorie abhängig von der minimalen Beinbreite ist, und somit ab etwa 45 cm alle Laserdatensätze korrekt erkannt werden, auf denen keine Person erkannt werden soll.

Die maximale Beinbreite hingegen beschreibt, wie breit ein Objekt sein darf, um als Bein detektiert zu werden. Dieser Wert steigt ab etwa 11 cm, also ungefähr bei dem Wert, der als minimale Beinbreite festgelegt wurde (siehe Abb. B.17). Durch stark faltige Hosen kann die Beinbreite jedoch stark von der eigentlichen Breite des Beines abweichen, wodurch sich der Klimax

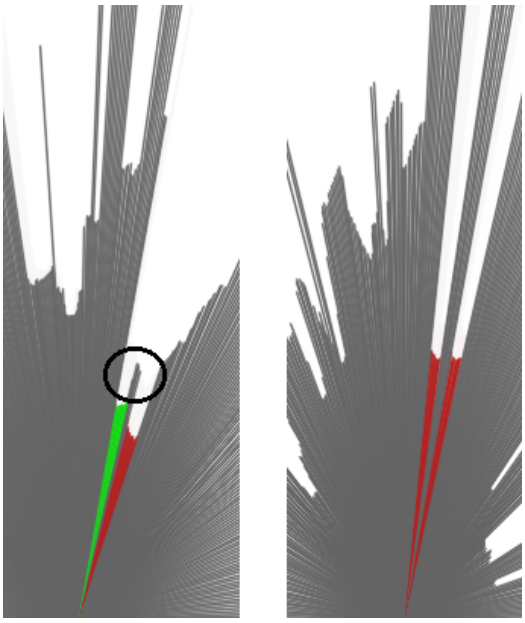


Abbildung 5.9: Beispiele für den Beinpaardetektor. Rot: Korrekt erkannt. Grün: Nicht erkannt. Links: Fehldetektion. Kreis wurde linkem Bein zugeordnet, deswegen nicht als Bein erkannt, weil Breite zu groß. Rechts: Korrekte Detektion in schwieriger Umgebung.

bei etwa 35 cm erklärt. Falls jedoch die maximale Beinbreite überschritten wird, aufgrund falsch zugeordneter Laserpunkte, wird ein Bein eventuell nicht detektiert (siehe Abb. 5.9). Da zudem etwa 60% Rockobjekte und 75% der unbekleideten Beine erkannt wurden, ist die korrekte Detektionsrate auf dem Datensatz ALL ab etwa 55 cm ein wenig höher.

Die Beintiefe lässt sich im Gegensatz zur Beinbreite nicht direkt aus den Laserdaten auslesen (siehe Abb. B.18). Sie verwendet den dem Roboter nächsten und weitesten Punkt eines Objektes zur Bestimmung der Tiefe und setzt diese dann in Relation zur Breite desselben. Dementsprechend wird der beste Wert für die minimale Beintiefe erreicht, wenn der Wert fast null ist und die positive Detektionsrate verschlechtert sich zunehmend mit der Größe des Parameters. Die Detektionsrate ist ab einem Wert von 0,75 für ALL besser als bei PAIR. Dies liegt daran, dass mit Zunahme der minimalen Beintiefe weniger falsche Beinpaare gefunden werden und somit einzelne Beine oder Personen mit Röcken, die vorher deswegen nicht gefunden werden konnten, nun detektiert werden.

Um die maximale Beintiefe zu bestimmen ist es ausschlaggebend, wie groß das Verhältnis zwischen Tiefe und Breite werden kann. Der optimale Wert wurde hier zwischen 1,76 und 1,96 (siehe Abb. B.19) gewählt. Bei einer Breite von 11,5 cm beträgt die maximale Tiefe demzufolge zwischen 20,24 und 22,54 cm. Wenn die maximale Beinbreite jedoch so niedrig ist, dass keine Beine detektiert werden können, dann werden insbesondere Rockobjekte erkannt, die ansonsten nicht gefunden worden wären.

Neben der Beinbreite und der Beintiefe ist die Beinform von Interesse zur Detektion eines Beines. Im Idealfall findet der Laser ein rundes Objekt und jeder Laserpunkt misst einen größeren Abstand je weiter er am Rand des Objektes misst. Dies ist im Normalfall nicht möglich. Gerade durch das Tragen von Hosen tritt hier eine Verzerrung auf, die eine annähernde Kreisform unmöglich macht. Jedoch nützt dieser Parameter Objekte auszuschließen, die flach oder konkav sind. Optimale Werte lieferte der Detektor für diesen Parameter bei einem Wert von 37 (siehe Abb. B.20). Die Detektionsrate wird bei höheren Werten schlechter, da das Objekt kreisrunder sein muss, je größer der Wert ist. Wenn der Wert kleiner ist, so ist die Detekionsrate geringfügig schlechter, da in diesem Fall andere Objekte als Beine erkannt wurden. Auch hier gilt, dass mit dem Sinken der Wahrscheinlichkeit ein Beinpaar zu detektieren andere Objekte mit höherer Wahrscheinlichkeit korrekt gefunden werden.

Das einzige Kriterium für die Bestimmung von Beinpaaren ist der maximale Abstand zwischen zwei detektierten Beinen. Dieser ist dadurch begrenzt, dass dieser aus anatomischen Gründe, bestimmte Werte nicht erreichen kann. Da es für den Beinpaardetektor jedoch reicht Personen in

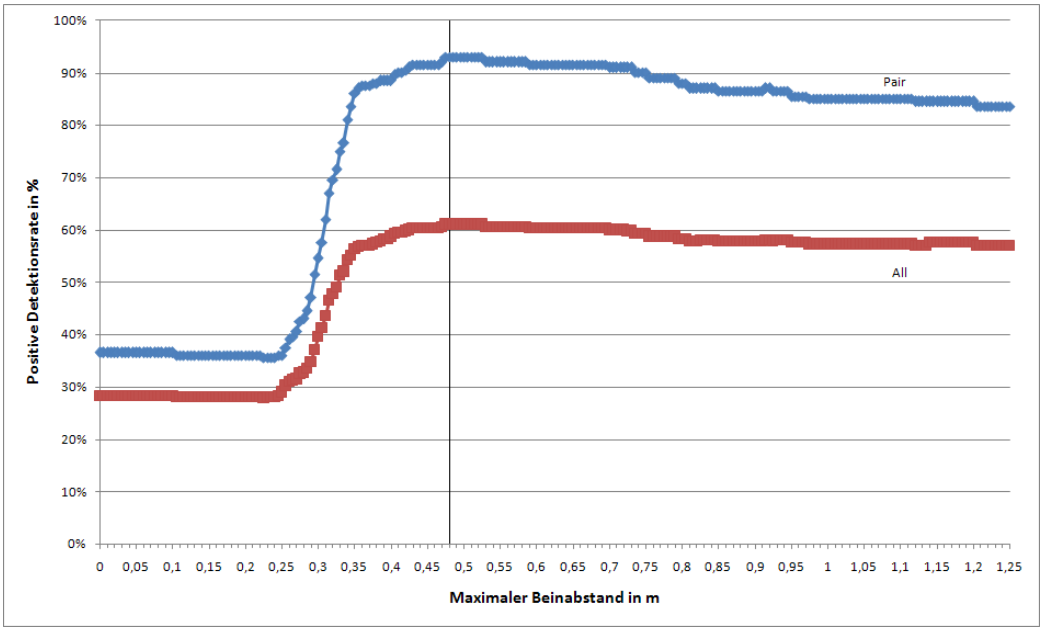


Abbildung 5.10: Verlauf bei Veränderung des Parameters LEGDIST. ◇ = Datensatz PAIR. □ = Datensatz ALL

natürlichen Posen zu erkennen kann dieser Wert weit niedriger gewählt werden, als es möglich wäre. Dadurch wird zudem vermieden, dass Beine verschiedener Personen demselben Beinpaar zugeordnet werden. Der Normalabstand liegt zwischen 25 und 35 cm, was einer entspannten Pose entspricht (siehe Abb. 5.10). Ein Wert von 48 cm war hier ausreichend, um als maximaler Abstand zwischen zwei Beinen zu gelten. Der Einfluss dieses Parameters auf andere Datensätze ist eher gering. Einzig unbekleidete Beine werden besser erkannt.

Abschließend lässt sich sagen, dass der Beinpaardetektor sehr gut funktioniert, wenn sich tatsächlich Beinpaare in seiner Umgebung befinden und nicht zu viel Ablenkung in seiner Umgebung ist. Dies sollte in einer Außenumgebung, für die CampusGuide vorgesehen war, noch besser funktionieren, da dort Ablenkungen wie Tisch-, Stuhlbeine oder Rechner nicht vorkommen. Damit sollte insbesondere die Detektionsrate steigen, wenn kein Beinpaar für den Laser-Scanner sichtbar ist.

5.3.2 Personendetektor

In diesem Abschnitt wird der komplette Personendetektionsmechanismus evaluiert. Dabei soll herausgestellt werden, wie gut das System als Gan-

zes funktioniert und wie die Laufzeit verringert werden kann mit minimalen Auswirkungen auf die Detektionsrate.

Vorbereitung Der Beinpaardetektor ist zwar schnell, lässt sich aber leicht, z. B. durch vorbeigehende Personen, ablenken. Aus diesem Grund wird er durch einen Gesichtsdetektor ergänzt, welcher an den berechneten Positionen nach Gesichtern sucht. Falls dies bestätigt wird, so fährt der Roboter zur entsprechenden Position.

In seiner ursprünglichen Form berechnet der Beinpaardetektor die maximal vier wahrscheinlichsten Positionen, an denen sich Personen befinden könnten, und weist ihnen Prioritäten zu. Diese Positionen werden dann vom Gesichtsdetektor nacheinander untersucht, wobei ein Erfolg verbucht wird, wenn auf mindestens drei von fünf Bildern eine Person zentral im Bild gefunden wurde. Falls er keinen Erfolg bei den vier ermittelten Positionen hat, dreht sich der Roboter jeweils um 120° und wiederholt den Vorgang bis er eine Person findet oder alle Richtungen geprüft hat.

Da die Bearbeitung eines Bildes durch den Gesichtsdetektor etwa 1,4 Sekunden dauert, kann die Personendetektion im schlimmsten Fall - bei drei Positionen mit vier Prioritäten und fünf Bildern - ungefähr 84 Sekunden dauern. Es offensichtlich, dass dieser Vorgang beschleunigt werden muss, da einer Person nicht zugemutet werden kann, so lange zu warten, bis der Roboter auf sie reagiert. Es soll im Folgenden also getestet werden, wie viele Prioritäten und wie viele Bilder ausreichen, um eine Person ausreichend gut und schnell zu finden.

Als Testdaten wurden 61 Datensätze im Ablaufszenario aufgenommen. Ein Testdatensatz besteht in diesem Fall aus allen vorgenommenen Messungen, die für die Personendetektion relevant sind. Dies sind im Einzelnen sämtliche aufgenommenen Bilder inklusive Detektion und die aufgenommenen Laserdatensätze mit untersuchten Prioritäten.

Um die Ergebnisse vergleichen zu können, wurden für jede Kombination aus Anzahl der Prioritäten (zwischen eins und vier) und die Anzahl der positiven Detektionen in Relation zu den untersuchten Bildern der Mittelwert über alle Datensätze bzgl. Detektionsrate und Zeit gebildet.

Testläufe Für jeden Testlauf wurde der Roboter in seinem Einsatzkontext getestet. Der Roboter wurde zu vorher ausgesuchten Orten berufen, an denen sich eine Testperson an einer vorher festgelegten Position befand, die sich in einem Radius von 5 m um den Zielpunkt des Roboters befand. Der Roboter musste der zu findenden Person also auf dem Weg zum Ziel eventuell ausweichen und sich drehen, um sie zu finden. Ziel war es, herauszufinden, ob

der Roboter die Person finden und anfahren würde und wie lange er dafür brauchen würde.

Zu diesem Zweck wurden für jeden Detektionsschritt und jedes aufgenommene und verarbeitete Bild die benötigte Zeit und der Erfolg der Detektion gemessen und gespeichert. Mit diesen Informationen wurde rekonstruiert, wie schnell und gut der Personendetektor gewesen wäre, wenn weniger Prioritäten des Beinpaardetektors und weniger Bilder durch den Gesichtsdetektor untersucht worden wären. Ziel war es hierbei zu bestimmen, wie schnell die Personendetektion als Gesamtsystem sein kann ohne große Einbußen in der Detektionsrate.

Auswertung Ausgangspunkt für die Personendetektion waren vier vom Beinpaardetektor ermittelten Positionen (PrioA (höchste) - PrioD (niedrigste)). Dabei musste auf drei von fünf untersuchten Bildern eine Detektion auf derselben Position erfolgen, damit ein Ziel ausgewählt und angesteuert wurde.

Dies dauerte im Durchschnitt 26,61 Sekunden, maximal 72.57 Sekunden und im besten Fall 7.14 Sekunden. Diese Zeiten sind dem Benutzer sowohl im Durchschnitt als auch im Maximalfall nicht zumutbar. Auch für den Minimalfall wäre eine schnellere Detektion wünschenswert. Die Detektionsrate lag für diese Parameterwahl bei 62,3% und ist noch verbesserungswürdig.

Das erste Ziel ist es nun zu ermitteln, welche Auswirkungen die Verringerung der Anzahl von Prioritäten auf das Ergebnis hat (siehe Tab. 5.4). Hier fällt auf, dass die durchschnittliche Zeit für die Personendetektion mit der Anzahl der betrachteten Prioritäten zunimmt, der Zeitzuwachs jedoch stets kleiner wird. Das liegt daran, dass in vielen Fällen entweder weniger Objekte gefunden werden, als untersucht werden können, oder dass der Personendetektor abbricht und auf das Ziel zufährt. Die maximal benötigte Zeit wächst relativ gleichmäßig mit der Anzahl der betrachteten Positionen. Die minimale Bearbeitungszeit ist in allen vier Fällen identisch, da stets die gleiche Anzahl an Bildern betrachtet wurde und in diesem Fall eine positive Rückmeldung des Gesichtsdetektors bereits nach der Untersuchung der ersten Position erfolgte. Die erste Erkenntnis hieraus ist also, dass die Verringerung der Anzahl der maximal zu untersuchenden Prioritäten, große zeitliche Einsparungen ermöglicht. Bezüglich der Detektionsraten fällt auf, dass diese mit der Anzahl der untersuchten Prioritäten wächst. Dabei ist die Rate, wenn nur die höchste Priorität untersucht wird, eklatant schlechter als bei PrioB. Diese ist jedoch nur geringfügig schlechter als bei PrioC oder PrioD. Dies liegt daran, dass nur ein Datensatz aus der Testmenge bei PrioB nicht erkannt wurde, weil die Person vom Beinpaardetektor nur an die dritte Stelle der

Liste gesetzt wurde. Hieraus lässt sich schlussfolgern, dass es durchaus Sinn macht nur die ersten zwei Prioritäten, die vom Beinpaardetektor berechnet werden, zu betrachten, da die Wahrscheinlichkeit ziemlich hoch ist, dass die Person mit diesen gefunden wird.

Genauer gesagt wurde die Person im Testlauf zu 85,25% innerhalb der ersten zwei Prioritäten eingeordnet und nur in 68,85% der Fälle in PrioA detektiert. Dies wirkt auf den ersten Blick eklatant schlechter als durch die Evaluation für den Beinpaardetektor ermittelt wurde, ist aber wie folgt begründet. Zum einen ist der Beinpaardetektor auf Beinpaare optimiert. Aus diesem Grund wurde der Fokus bei den Testläufen der Personendetektion auf natürliche Beinstellungen gelegt und davon ausgegangen, dass nicht jede Person direkt dem Roboter zugewandt ist. Dadurch wurde die Person oftmals nur als Bein oder Rockobjekt erkannt und dadurch niedriger eingestuft, was zur Folge hatte, dass andere Positionen höher eingestuft und somit vorher untersucht wurden. Zum anderen brach die Personendetektion in einigen Fällen ab, bevor sie die Person erfassen konnte. Dies lag entweder daran, dass der Gesichtsdetektor bereits an anderer Stelle angeschlagen hatte oder der Beinpaardetektor die Person nicht finden konnte, weil sie zu weit weg war oder zu nah vor einer Wand stand.

Prio	Pos. Bild	# Bild	∅ Zeit	Max Zeit	Min Zeit	∅ Rate
PrioD	3	5	26,61	72,57	7,14	62,3%
PrioC	3	5	23,9	58,28	7,14	62,3%
PrioB	3	5	20,17	43,9	7,14	60,7%
PrioA	3	5	13,14	22,42	7,14	47,5%

Tabelle 5.4: Einfluss der Anzahl der vom Gesichtsdetektor betrachteten Prioritäten

Mittels Reduktion der betrachteten Prioritäten konnte die Zeit für die Personendetektion im Schnitt um etwa 6 Sekunden reduziert werden, bei etwa 1,5%-tigen Einbußen bei der Detektionsrate, wobei der Vorgang in keinem Fall länger als 45 Sekunden dauerte. Dies ist zwar deutlich besser als in der ursprünglichen Version bzgl. der benötigten Zeit, jedoch ist noch weiteres Beschleunigungspotenzial vorhanden, wenn die Anzahl der vom Gesichtsdetektor untersuchten Bilder reduziert wird.

Hierbei gibt es zwei Faktoren, die beeinflusst werden können. Dies ist zum einen die Laufzeit, die von der Anzahl der betrachteten Bilder abhängt, und zum anderen die Detektionsrate, die von der Anzahl der positiven Detektionen abhängt, die ausreichend ist, damit der Gesichtsdetektor eine positive Antwort gibt. Es gilt demzufolge einen Kompromiss zu finden, der beide Ziele so gut wie möglich erfüllt.

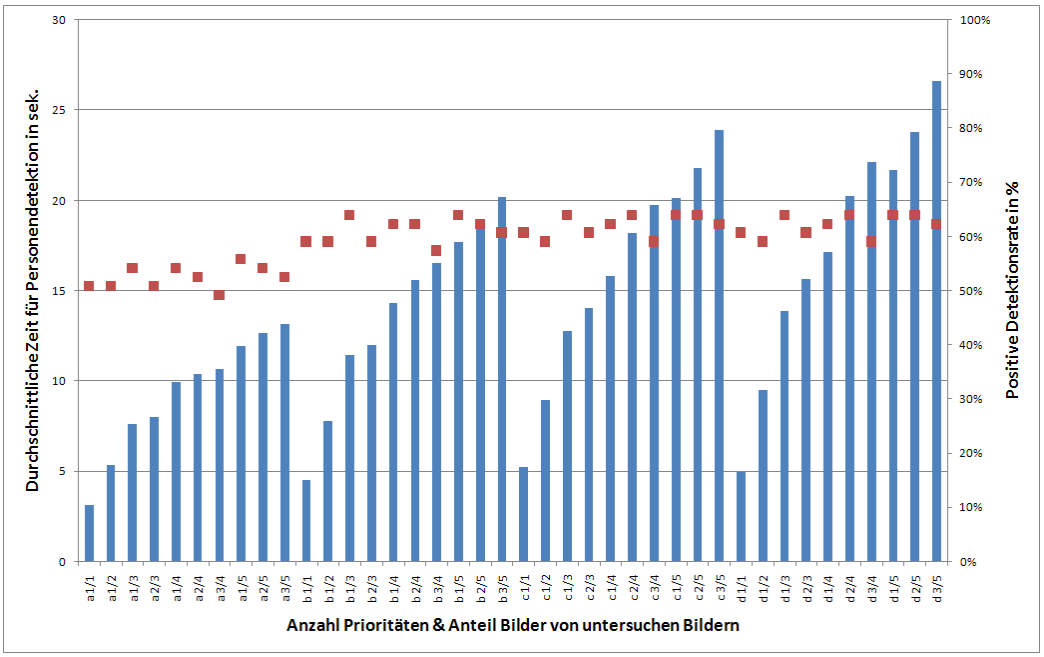


Abbildung 5.11: Detektionsrate (Punktdiagramm) & Zeitbedarf (Balkendiagramm) unter Betrachtung verschiedener Parameter (Annotation: x y/z (wobei $x \in \{a, b, c, d\}$ (Priorität), $z \in \{1 - 5\}$ (Anzahl betrachtete Bilder), $y \in \{1 - 5\}$ und $y \leq z$) (Anzahl positive Detektionen)

.

Nach Messung der verschiedenen Situationen ergaben sich Zeiten und Detektionsraten wie in Abbildung 5.11. Das Erste, das hier ins Auge fällt, ist die Bestätigung, dass die Verwendung der ersten zwei Prioritäten ausreichend ist. Aus diesem Grund sind im Folgenden nur die Daten in Tabelle 5.11 mit b y/z) relevant. Logischerweise wächst der zeitliche Bedarf mit der Anzahl der betrachteten Bilder. Zudem wächst dieser auch, je höher der Anteil an Bildern ist, die positiv erkannt werden müssen, damit der Personendetektor eine positive Antwort gibt. Dies liegt daran, dass hierdurch die Wahrscheinlichkeit wächst, dass eine Abweisung erfolgt, wodurch weitere Positionen untersucht werden, was dann in einer Verlängerung der Durchschnittslaufzeit resultiert. Aus Tabelle 5.5 wird ersichtlich, dass die Betrachtung von vier bzw. fünf Bildern im schlimmsten Fall länger als eine halbe Minute dauert. Dies erachten wir als eine zu lange Wartezeit für die Person, weshalb wir diese Parametersätze für ungeeignet halten. Eine Detektionsrate von mehr als 60% ist wünschenswert, weshalb unter diesen Gesichtspunkten nur die Betrachtung von 3 Bildern, bei einer positiven Detektion auf diesen, Sinn macht. Jedoch liegt die niedrigere Detektionsrate bei zwei von drei Bildern daran, dass in diesen Fällen der Gesichtsdetektor nur auf einem der ersten drei Bilder die Person erkannt hat. In anderen Fällen fuhr der Roboter zu einer Position, da er einer Position, an der sich kein Gesicht befand ein ebensolches fälschlicherweise detektierte. Zur Verifikation, ob sich an einer Position ein Mensch befindet, ist es aber durchaus sinnvoll eine Detektionsrate von mehr als 50% zu haben, weshalb wir uns entschieden haben, dass zwei von drei untersuchten Bildern mit positiver Detektion sowohl hin- als auch ausreichend sind, um eine Person zu finden. Zudem ist es wünschenswert, dass der Roboter entweder zur Person fährt oder stehen bleibt, wenn er sie nicht finden kann, da die Anfahrt eines falschen Zieles eine negativere Reaktion bei der Kontaktperson hervorrufen würde, als wenn der Roboter stehen bleibt und sich grafisch dafür entschuldigt, dass er die Person nicht finden konnte.

Zuletzt ist noch interessant, warum und in welchen Situationen die Personendetektion richtige oder falsche Ergebnisse liefert. Besonders in störungsfreien Umgebungen, wenn die Zielperson dem Roboter zugewandt ist und die Beine klar voneinander unterscheidbar sind, wird die Zielperson sehr schnell und zuverlässig gefunden (siehe Abb. 5.12). Auch wenn dies nicht der Fall ist, ist die Detektion meist zuverlässig (siehe Abb. B.21). Probleme traten hingegen meistens auf, wenn die Zielperson in ungünstiger Stellung für die Beinpaardetektion stand und zudem viele Störungen im Sichtbereich des Laser-Scanners waren (siehe Abb. 5.13). Problematisch war zudem, wenn der Gesichtsdetektor die Person aufgrund von Lichtverhältnissen nicht finden konnte, der Beinpaardetektor jedoch in der Richtung der Zielperson weitere Objekte fälschlicherweise als Person detektierte. Wenn sich die Person des-

Prio	Pos. Bild	# Bild	∅ Zeit	Max Zeit	Min Zeit	∅ Rate
PrioB	1	1	4,53	10,31	1,73	59,02%
PrioB	1	2	7,78	18,72	3,11	59,02%
PrioB	2	2	8,3	18,72	3,11	55,74%
PrioB	1	3	11,19	27,15	4,45	63,93%
PrioB	2	3	11,98	27,15	4,45	59,02%
PrioB	3	3	12,63	27,15	4,45	50,82%
PrioB	1	4	14,32	35,58	5,38	62,3%
PrioB	2	4	15,57	35,58	5,38	62,3%
PrioB	3	4	16,52	35,58	5,38	57,38%
PrioB	4	4	16,52	35,58	5,38	45,9%
PrioB	1	5	17,69	43,9	7,14	63,93%
PrioB	2	5	18,75	43,9	7,14	62,3%
PrioB	3	5	20,17	43,9	7,14	60,66%
PrioB	4	5	20,4	43,9	7,14	52,46%
PrioB	5	5	20,4	43,9	7,14	42,62%

Tabelle 5.5: Vergleich der Ergebnisse des Gesichtsdetektors. Auswirkungen unterschiedlicher Anzahlen von Bildern und positiven Detektionen auf diesen (hier für PrioB)

halb im Sichtbereich der Kamera befand und somit der Gesichtsdetektor eine positive Antwort gab, fuhr der Roboter zum falschen Ziel (siehe Abb. B.22). Weitere Beispiele werden in Abbildung 5.14 dargestellt.

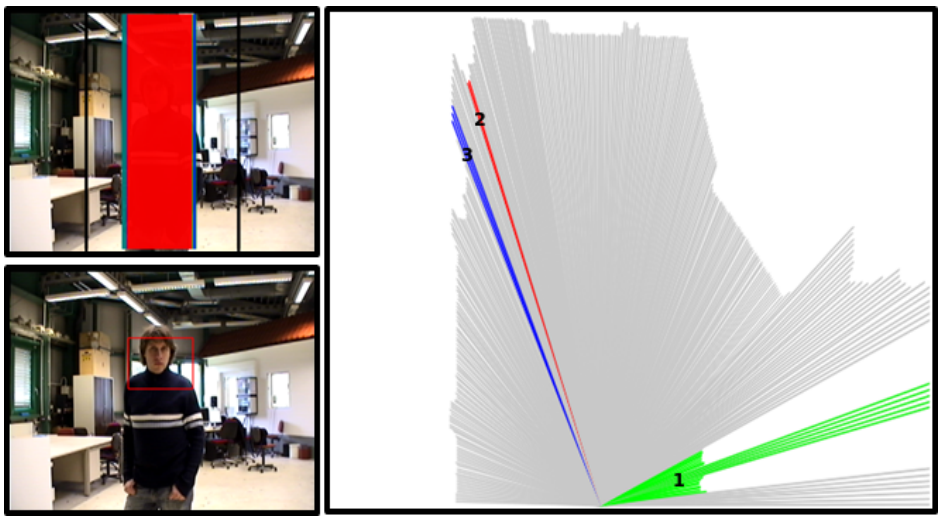


Abbildung 5.12: Positives Beispiel 1. Die Person wurde rechts vom Roboter als Beinpaar erkannt (der Bereich zwischen den Beinen wird hierbei dem Beinpaar zugeordnet) und auf fünf von fünf Bildern vom Gesichtsdetektor erkannt (1 = Beinpaar (Person/Ziel). 2 = Rockobjekt. 3 = Rockobjekt mit niedrigerer Priorität)

.

Schlussfolgernd lässt sich sagen, dass viele der auftretenden Probleme in einem Außenszenario nicht auftreten würden. Die Detektionsrate sollte dort also größer sein. Zielkoordinaten für einzelne Ziele könnten für die Datenbank so gewählt werden, dass möglichst wenig Störungen die Ergebnisse des Beinpaardetektors beeinflussen, und dass die Person nicht die Gefahr läuft, zu nah an einem Hindernis zu stehen. Durch bessere Lichtverhältnisse als im Innenszenario sollte zudem die Wahrscheinlichkeit sinken, dass der Gesichtsdetektor eine Person nicht detektiert. Durch die Verwendung der durch den Laserdetektor berechneten Positionen und Abstände zum Roboter könnte zudem der Suchraum für die Kamera weiter eingeschränkt werden. Somit sollte der hier verwendete Personendetektor in Außenarealen deutlich besser funktionieren.

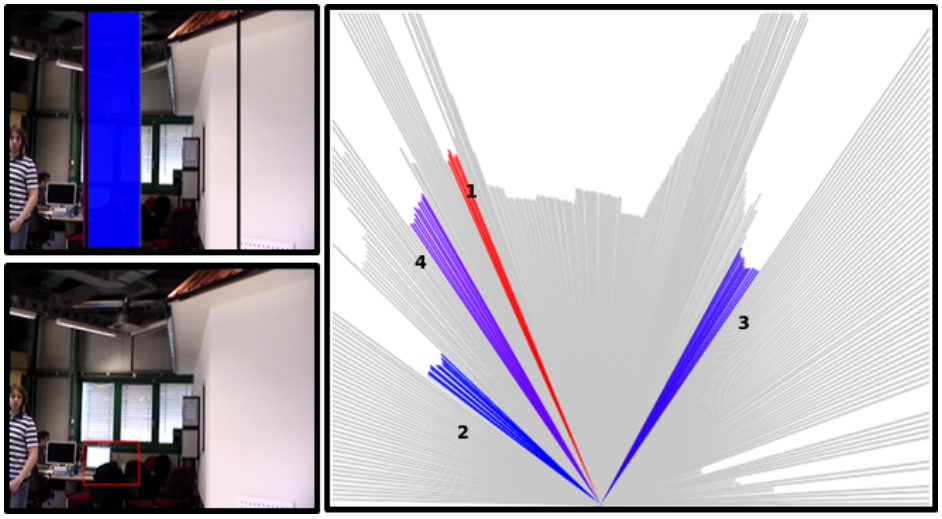


Abbildung 5.13: Negatives Beispiel 1. Ein Objekt wurde fälschlicherweise als Bein klassifiziert, die Person wegen der Beinstellung als Rockobjekt klassifiziert und aufgrund des Abstands zum Roboter auf Priorität zwei gesetzt (1 = Bein (Ziel). 2 = Rockobjekt (Person). 3-4 = Rockobjekt)

.

5.4 Gesamtsystem

Eine Evaluation des Gesamtsystems ist schwer zu realisieren, da in dieser sehr viele Faktoren zum tragen kommen. Es muss sowohl das Zusammenspiel der einzelnen Komponenten als auch die Benutzbarkeit der graphischen Oberflächen erfasst werden. Die Benutzbarkeit des Systems lässt sich daher am besten durch eine Benutzerstudie erfassen. Diese ist zwar subjektiv gefärbt, kann aber dennoch Aufschluss über die Stärken und Schwächen des Systems geben.

5.4.1 Aufbau und Durchführung

Wir haben eine Benutzerstudie mit zwölf Probanden in einer Testumgebung im Institut für Roboterforschung durchgeführt. Die Teilnehmer waren nicht an der Projektgruppe beteiligt und hatten keine Gelegenheit sich vorher mit CampusGuide vertraut zu machen. Für die Lokalisation in der Testumgebung wurde eine Karte wie in Kap. 4.3.1 beschrieben erzeugt. Für die Pfadplanung wurde ein Graph auf Basis dieser Karte manuell erstellt. Auf diesem Graphen wurde einige Orte als Navigationsziele festgelegt.

Die Aufgabe der Probanden bestand darin, CampusGuide mit einer SMS

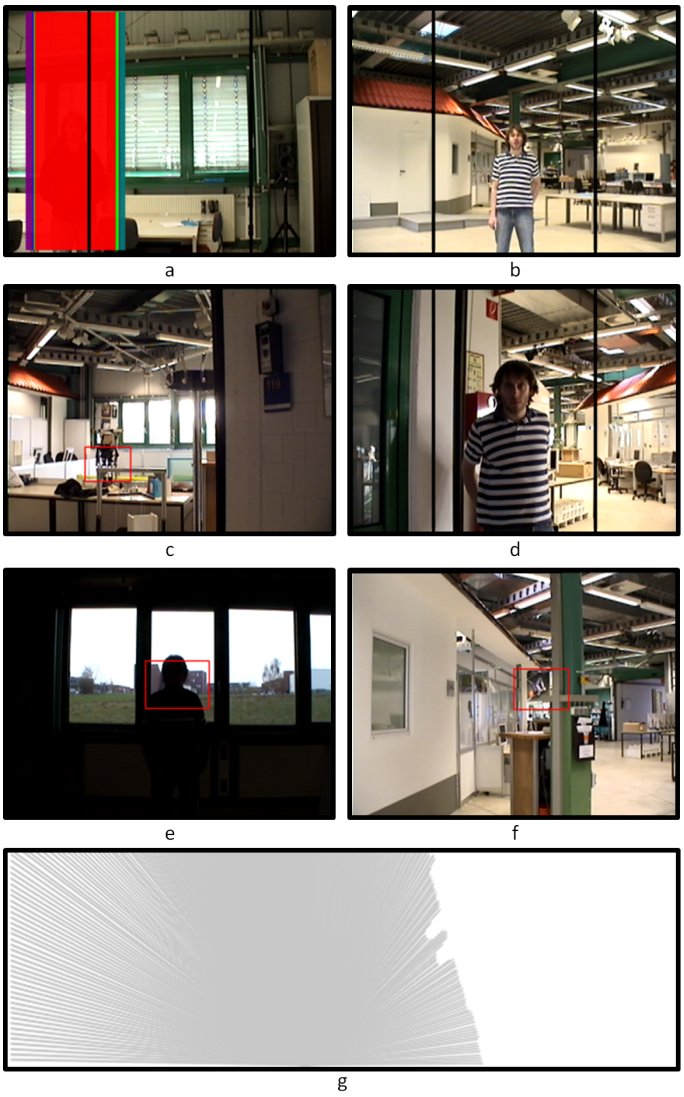


Abbildung 5.14: Weitere Beispiele. a) Die Person befindet sich außerhalb des interessanten Bereichs am Bildrand und wird korrekterweise nicht angefahren. b) Die Person befindet sich zu weit vom Roboter weg und ist nicht gut genug vom Hintergrund separierbar. c) Ein Roboterskelett wird fälschlicherweise als Person detektiert und angefahren. d) Die Zielperson wird aufgrund der Lichtverhältnisse vom Gesichtsdetektor nicht gefunden. e) Anhand der Silhouette wird die Person korrekt detektiert. f) Ein falsches Ziel wird vom Gesichtsdetektor gefunden und angefahren, bevor die Zielperson erfasst wurde. g) Die Person befindet sich zu nah an der Wand und wird deshalb vom Beinpaardetektor nicht als eigenes Objekt wahrgenommen.

.

zu rufen, sich zu mehreren Zielen führen zu lassen und danach den Roboter nach Hause zu schicken. Danach sollten sie einen Fragebogen (siehe Abb. C.5) ausfüllen um die Benutzbarkeit des Systems zu bewerten. Zu Beginn des Tests wurde jedem Probanden ein Informationsbogen (siehe Abb. C.4) ausgehändigt, der die durchzuführenden Aufgaben zusammenfasst.

Um einen Test der Hindernisvermeidung zu ermöglichen wurden Hindernisse an festen Positionen auf dem Weg des Roboters platziert. Es wurde sichergestellt, dass der Roboter in jedem Testlauf einem Hindernis ausweichen musste.

5.4.2 Ergebnisse

Auf dem Evaluationsbogen sollten die Benutzer ausgewählte Funktionen von CampusGuide auf einer Skala von 1 (schlecht) bis 5 (gut) bewerten. Bei den in diesem Kapitel angegebenen Werten handelt es sich, wenn nicht anders angegeben, um die Mittelwerte der Antworten aller Probanden. Die komplette Auswertung des Fragebogens ist in Anhang C.3 zu finden.

Das Rufen des Roboters mittels einer SMS wurde von den Benutzern als einfach empfunden (⌀ 4,17). Dies ist möglicherweise darauf zurückzuführen, dass in der Testumgebung nur wenige Zielpunkte verfügbar waren. In einem Außenareal ohne klar markierte Punkte könnten hier Schwierigkeiten auftreten. Eine Möglichkeit dieses Problem zu umgehen sehen wir darin, das Rufen des Roboters über eine Applikation für Smartphones zu realisieren, die automatisch auf die GPS-Daten des Benutzerhandys zugreifen kann. Dies würde es auch ermöglichen, dem Benutzer mehr Rückmeldung von Seiten des Roboters zu geben. Die Antwort-SMS, die dem Benutzer vom Roboter zugeschickt wurde, empfanden die Probanden zwar als hilfreich (⌀ 4,5), dennoch könnten zusätzliche Informationen an dieser Stelle hilfreich sein. Eine Möglichkeit dafür wäre z. B. eine erwartete Ankunftszeit.

Beim Anmelden des Benutzers am Roboter traten keinerlei Probleme auf. Sowohl der Aufbau der Eingabemaske, als auch Länge und Zusammensetzung des Passworts erfüllen damit unsere Anforderungen.

Die Zielauswahl über den Touchscreen wurde von den Benutzern mit 3,42 bewertet. Einige Probanden erkannten nicht sofort, dass sie bereits eine Eingabe hätten tätigen können. Dies deutet darauf hin, dass die Benutzerführung durch die Auswahlmaske nicht ausreichend war.

Als großer Schwachpunkt stellte sich die Geschwindigkeit des Roboters heraus. Die Mehrzahl der Benutzer (75%) empfand die Geschwindigkeit als zu niedrig. Einzelne Benutzer wünschten sich eine Möglichkeit auf die Geschwindigkeit des Roboters Einfluss nehmen zu können. Dies ließe sich gut in eine Smartphone-Applikation zum Rufen des Roboters integrieren.

Die Hindernisvermeidung von CampusGuide wurde mit 3,83 bewertet. Dies geht möglicherweise darauf zurück, dass den Benutzern die internen Abläufe der Hindernisvermeidung nicht vertraut waren. Dass der Roboter auch Zielen ausweicht, die nicht direkt seinen Pfad blockieren und enge Durchgänge, z. B. Türrahmen, meidet könnte von den Benutzern als überflüssig empfunden worden sein. Dieses Verhalten ist für eine sichere Hindernisvermeidung jedoch absolut notwendig.

Die Emotionen des Roboters wurden mit 3,75 bewertet. Dennoch hat sich die Entscheidung CampusGuide mit Emotionen auszustatten als richtig herausgestellt, da $^2/_3$ der Benutzer absichtlich versucht haben den Pfad des Roboters zu blockieren, um seine Reaktion zu testen. Der Wiederholung eines solchen Verhaltens sollen die Emotionen entgegenwirken.

Ein Proband hat den Roboter bei der Führung kurzzeitig verloren und musste sich beeilen um diesen wieder einzuholen. Dies konnte passieren, weil die Bluetooth-Abstandsmessung, die solche Situationen verhindern soll, zum Zeitpunkt der Nutzerstudie noch nicht einsatzbereit war. Zusätzlich hat sich gezeigt, dass eine Möglichkeit für den Benutzer die Führung zu unterbrechen wünschenswert wäre.

In den meisten Fällen (10 von 12 Probanden) traten während der Benutzung von CampusGuide keine störenden Wartezeiten auf. In den Fällen wo Verzögerungen beobachtet wurden gingen diese auf die Personendetektion zurück, die selbst unter guten Bedingungen oft $\sim$30 Sekunden benötigte um einen möglichen Benutzer zu identifizieren.

Die Schritte bei denen die meisten Probleme auftraten waren das Rufen des Roboters per SMS und die Anmeldung am Roboter. Der vom Benutzer per SMS gesendete Zielpunkt wurde ohne Fehlertoleranz mit den Namen der möglichen Zielpunkte in der Datenbank verglichen. Insbesondere Groß- und Kleinschreibung stellten die Benutzer dabei vor Probleme. Durch die Verwendung eines toleranteren Vergleichs, z. B. unscharfe Textsuche, kann dieses Problem gelöst werden. Die Probleme bei der Anmeldung entstanden dadurch, dass die Suche nach dem Benutzer nicht abgebrochen werden konnte. Ein Proband der versuchte, sich am Roboter anzumelden obwohl die Personendetektion noch nicht beendet war konnte in die Situation geraten, dass das CampusGuide plötzlich zu einer anderen Person fuhr, obwohl der Proband sich bereits angemeldet hatte. Dieses Problem kann durch den Abbruch der Personendetektion bei erfolgreicher Anmeldung gelöst werden.

Die Benutzbarkeit des Gesamtsystems wurde von den Probanden durchschnittlich mit 3,92 bewertet. Obwohl die Nutzerstudie in einer kontrollierten Testumgebung durchgeführt wurde lieferte sie wichtige Hinweise für die weitere Verbesserung von CampusGuide. Zusätzlich erlaubt uns der Austausch mit den Probanden besser einzuschätzen, welche Schritte der Führung für

einen Benutzer, der nicht mit den internen Abläufen des Systems vertraut ist, flüssiger gestaltet werden könnten.

Kapitel 6

Zusammenfassung und Ausblick

Mit CampusGuide ist es uns gelungen, einen mobilen Roboter zu entwickeln, der autonom Personen zu einem gewünschten Ziel führen kann. Zu diesem Zweck wird CampusGuide mittels einer SMS kontaktiert, welche er mit einem Passwort beantwortet und dann zur Position des Nutzers fährt. Den Weg zum Nutzer plant CampusGuide auf einem vorher festgelegten Graphen. Auf dem berechneten Weg weicht er lokalen Hindernissen aus und berechnet befahrbare Wege. Sobald CampusGuide am Ziel angekommen ist, sucht er nach der Zielperson und fährt zu dieser. Daraufhin kann der Nutzer sich mittels des übermittelten Passwortes anmelden und ein Ziel wählen, wohin ihn CampusGuide daraufhin geleitet. Sobald dieses Ziel erreicht ist, kann der Nutzer sich ein neues Ziel aussuchen oder sich abmelden, wenn er an seinem finalen Ziel angekommen ist. Dies wurde im Zuge unserer Nutzerstudie von den Teilnehmern durchgehend positiv bewertet.

Zusätzlich zu den beschriebenen Funktionen wurden weitere Funktionalitäten entwickelt, die auf den Außeneinsatz ausgelegt sind und deshalb im System nicht zum Einsatz kommen, da der Roboter nur in Innenbereichen zuverlässig funktioniert.

Für CampusGuide 2.0 wäre es also sinnvoll einen Roboter zu verwenden, der in Außenbereichen problemlos funktioniert. Somit könnten sowohl bereits von uns entwickelte als auch gänzlich neue Funktionalitäten hinzugefügt werden.

Zu ersteren Funktionen gehört die Navigation auf einer mittels OSM-Daten berechneten Karte, die dann zeitgleich über die Webapplikation grafisch angezeigt werden könnte. Wenn der Roboter in Außenbereichen verwendet wird lässt sich außerdem die Untergrunderkennung integrieren, die bereits auf dieses Szenario optimiert wurde. Damit der Benutzer während einer Führung nicht den Anschluss an den Roboter verliert haben wir eine Bluetooth-Abstandsmessung entwickelt, welche jedoch in CampusGuide

nicht zum Einsatz kommt.

Zudem gibt es natürlich noch viele weitere Anwendungen und Funktionen, die CampusGuide sinnvoll ergänzen würden. Zum einen wäre es sehr nutzerfreundlich dem Benutzer eine Smartphone-Applikation zur Verfügung zu stellen, mit Hilfe derer er CampusGuide rufen könnte, und sich somit nicht mehr selbst lokalisieren müsste, da die aktuelle Position automatisch übermittelt werden könnte. Außerdem würde durch solch eine Applikation auch das Bluetooth-Tracking erleichtert. Zusätzlich wäre es denkbar, dass Nutzer den Roboter reservieren könnten, so dass er sie, z.B. wenn sie mit der S-Bahn ankommen, zu einer bestimmten Uhrzeit an einem gewünschten Ort abholt. Da der Roboter auf dem Weg zurück zur Ladestation unbenutzt ist, wäre es sinnvoll, wenn verirrte Personen CampusGuide anhalten könnten und ihn ohne Handyruf benutzten könnten. Ein weiteres denkbares Szenario wäre, dass ein Nutzer noch in der Mensa essen will, bevor er zu seinem Ziel gebracht werden will und somit dem Roboter mitteilen könnte, vor der Mensa auf ihn zu warten, bevor die Reise fortgesetzt wird. Im Sinne der Benutzerfreundlichkeit wäre es zudem wünschenswert, wenn CampusGuide mittels Sprachausgabe dem Benuter wichtige Informationen übermitteln würde. In diesem Zusammenhang wäre es sogar möglich CampusGuide zusätzlich Fremdenführer-Funktionen durchführen zu lassen, indem eine Karte des Campus auf dem Bildschirm angezeigt wird, auf denen markante Punkte angezeigt werden, und CampusGuide sprachlich dem Nutzer bestimmte Informationen über bestimmte Orte mitteilt. Zudem wäre es für den Nutzer hilfreich, wenn er nicht nur zum Zielgebäude gebracht, sondern auch hineingeleitet würde. Auch auf technischem Level gibt es noch einige Ideen, die CampusGuide verbessern würden. Zum einen könnte mittels eines Infrarotsensors Höhenunterschiede im Gelände festgestellt werden, um zu vermeiden, dass CampusGuide Treppen hinunterfährt. Zum anderen wäre der Einsatz eines 3D-Laserscanners sinnvoll, um Personen besser zu finden und zu lokalisieren.

Bei CampusGuide handelt es sich jetzt schon um eine gut funktionierende Möglichkeit, Personen autonom zu einem Ziel zu führen. Mittels der vorgeschlagenen Erweiterungen ist es denkbar, dass CampusGuide 2.0 tatsächlich eingesetzt werden könnte, um Gäste auf dem Campus autonom über diesen zu führen.

Anhang A

Vollständige Liste der Zustandscodes

Roboterzustand		
Wert	Zustand	Beschreibung
0	Idle	Der Roboter ist einsatzbereit
1	Busy	
2	Moving to Client	Der Roboter fährt zu einem Benutzer
3	At Client Pos	Der Roboter hat die von User angegebene Position erreicht
4	Moving to Target	Der Roboter fährt ein vom User gewähltes Ziel an
5	At Target	Das vom User angegebene Ziel wurde erreicht
6	Moving Home	Nach beendeter Navigation fährt der Roboter seine Ladestation an
7	Maintenance Mode	Wartungsmodus. Der Roboter nimmt keine Anfragen von Usern an
8	Error Mode	Ein Fehler ist aufgetreten, die genaue Art des Fehlers ist im Datenfeld Fehlercode gespeichert
9	Moving to exact user pos	Benutzer wird mit Personenerkennung gesucht
10	At exact user pos	Benutzer wurde identifiziert und interagiert

Tabelle A.1: Die Werte die das Datenfeld Roboterzustand im Blackboard annehmen kann mit ihrer Bedeutung.

Navigations-Zustand		
Wert	Zustand	Beschreibung
0	Idle	
1	Calculating Path	
2	Path calculated	
3	Moving	
4	Target Reached	
5	Error	

Tabelle A.2: Die Werte die das Datenfeld NAV-Zustand im Blackboard annehmen kann mit ihrer Bedeutung.

Comm Zustand		
Wert	Zustand	Beschreibung
0	Idle	
1	Busy	
2	Requesting PSW and GPS Path Calc	
3	Requesting Home Calc	
4	Requesting GPS Calc	
5	Requesting PSW	
6	Requesting Start Journey	
7	Error	

Tabelle A.3: Die Werte die das Datenfeld Comm-Zustand im Blackboard annehmen kann mit ihrer Bedeutung.

GUI Zustand		
Wert	Zustand	Beschreibung
0	Idle, Ready for Input	
1	Waiting for User	
2	User making Input	
3	Requesting Home Calc	
4	Requesting GPS Calc	
5	Requesting Start Journey	
6	Error	
7	Requesting move to user	

Tabelle A.4: Die Werte die das Datenfeld GUI-Zustand im Blackboard annehmen kann mit ihrer Bedeutung.

Login Zustand		
Wert	Zustand	Beschreibung
0	Open (Ready for input)	
1	Locked till User PSW	
2	Locked (Admin PSW rqd)	
3	User Logged In	
4	Admin logged in	
5	Error	
6	-	Keine Funktion
7	-	Keine Funktion

Tabelle A.5: Die Werte die das Datenfeld Login-Zustand im Blackboard annehmen kann mit ihrer Bedeutung. Ob ein User/Admin eingeloggt ist bzw. die GUI für Eingaben gesperrt ist

Fehlercodes		
Bit	Beschreibung	Geändert durch:
15	-	Nicht verwendet
14	Error during PSW generation	Comm
13	GPS invalid	Nav
12	Path calculation failed	Nav
11	Stuck (not because of obstacles)	Nav
10	Surrounded by Obstacles	Nav
9	Path to Target blocked	Nav
8	Path blocked, Waiting	Nav
7	Maintenance Mode	GUI/Comm
6	Battery Low	GUI
5	Battery Charging	GUI
4	Error after Path Calculation	GUI/Comm
3	Error during Path Calculation	GUI/Comm
2	GUI not responding	Nav/Comm
1	Comm not responding	GUI/Nav
0	Nav not responding	GUI/Comm

Tabelle A.6: Die Werte die das Datenfeld Fehlercode im Blackboard annehmen kann mit ihrer Bedeutung.

Anhang B

Bilder

B.1 GUI

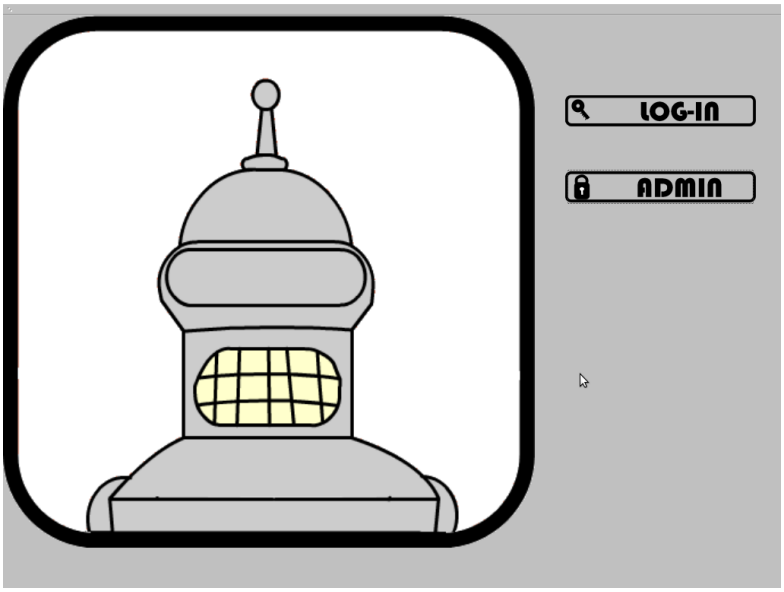


Abbildung B.1: Hauptformular

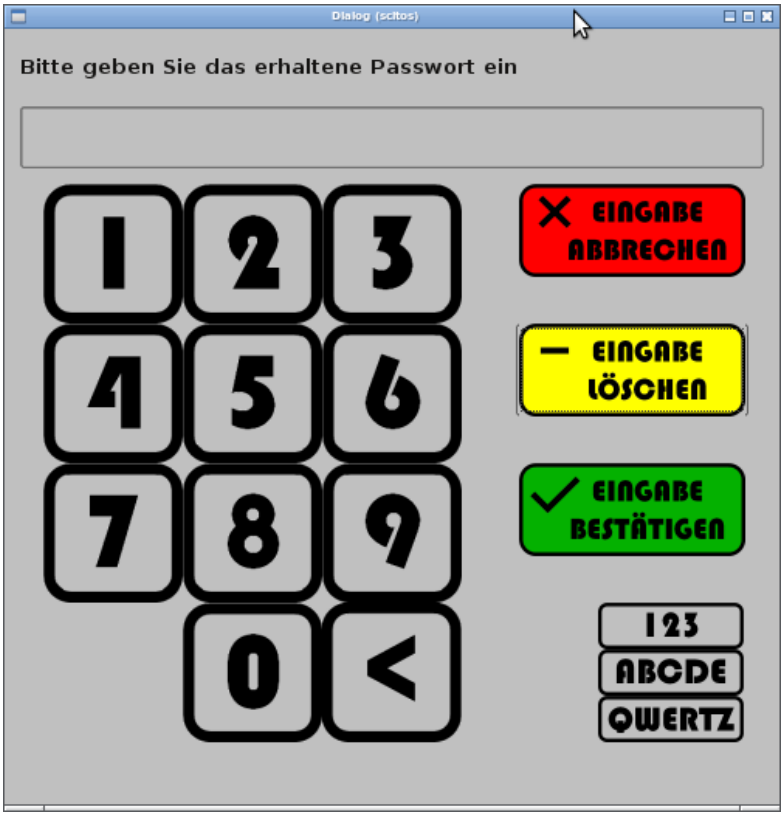


Abbildung B.2: Eingabemaske (NumPad)

Abbildung B.3: Eingabemaske (QWERTZ)

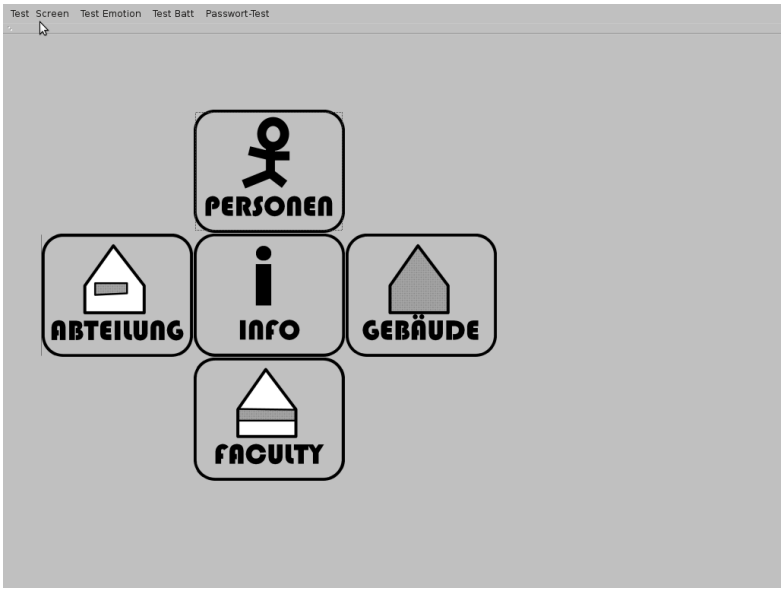


Abbildung B.4: Listenauswahl

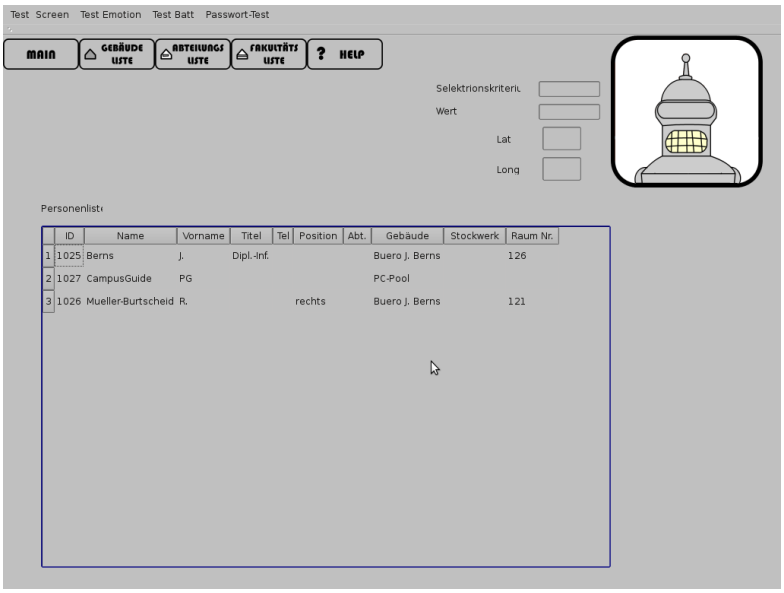


Abbildung B.5: Zielauswahlliste Personen

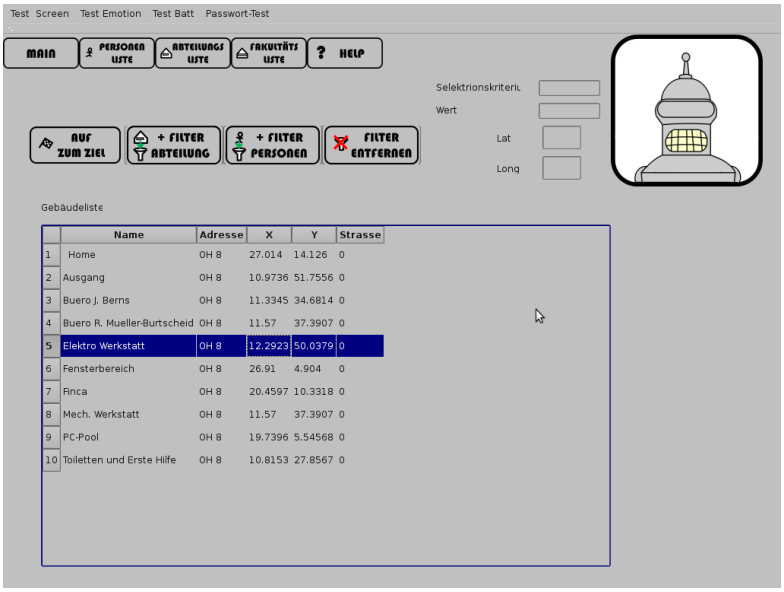


Abbildung B.6: Zielauswahlliste Gebäude, selektierter Eintrag

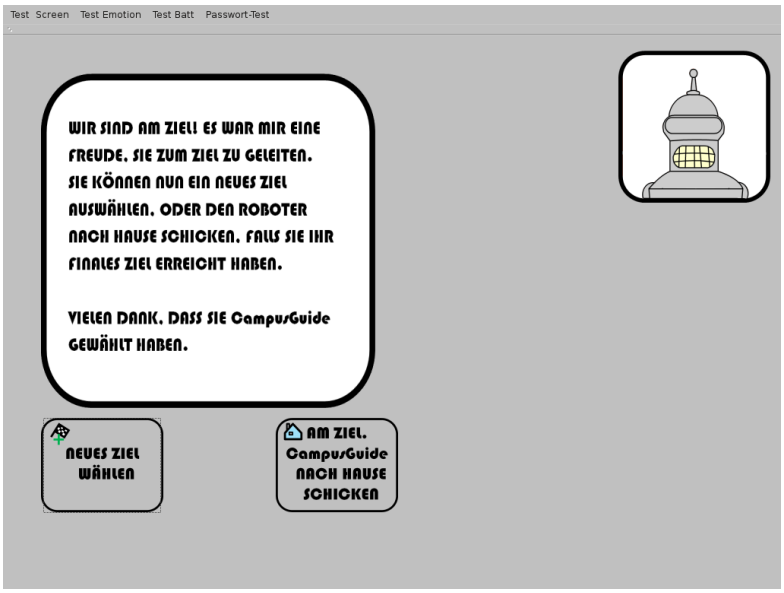


Abbildung B.7: „Ziel erreicht“ - Bildschirm

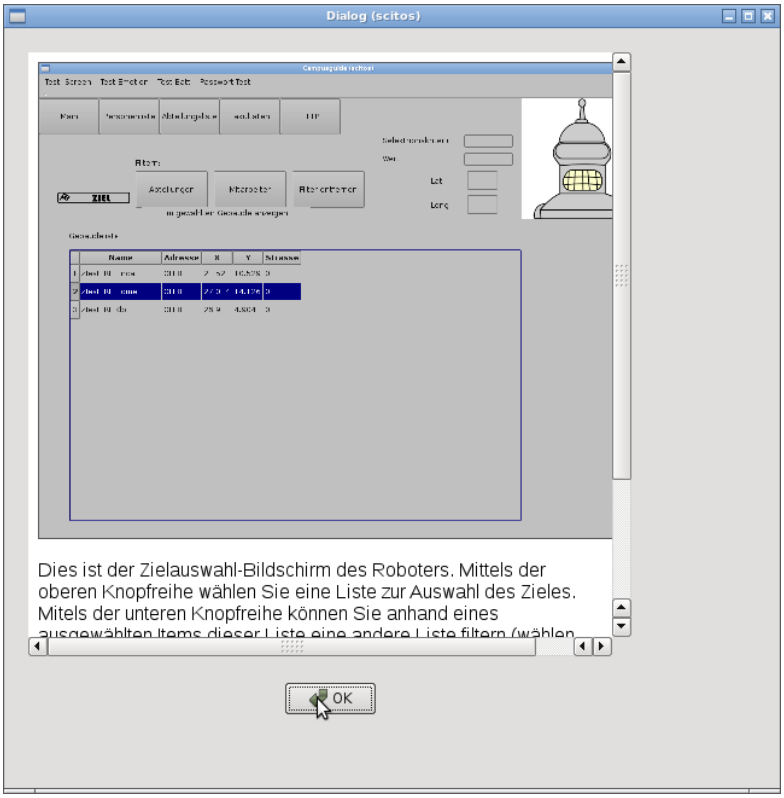


Abbildung B.8: Hilfebildschirm

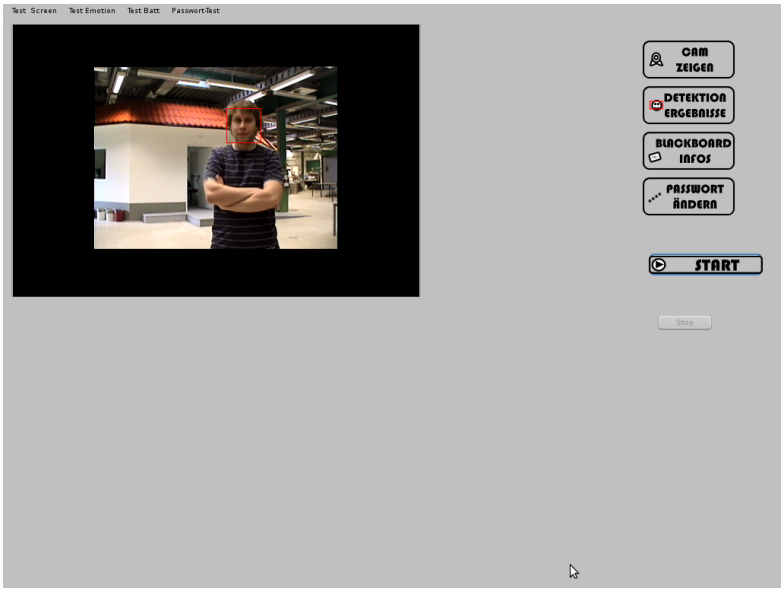


Abbildung B.9: Adminbildschirm: Kamera

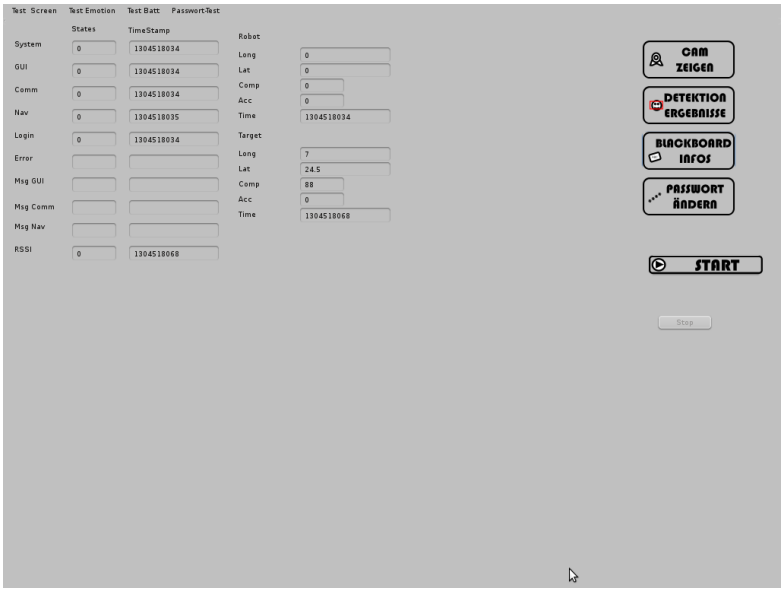


Abbildung B.10: Adminbildschirm: Blackboard

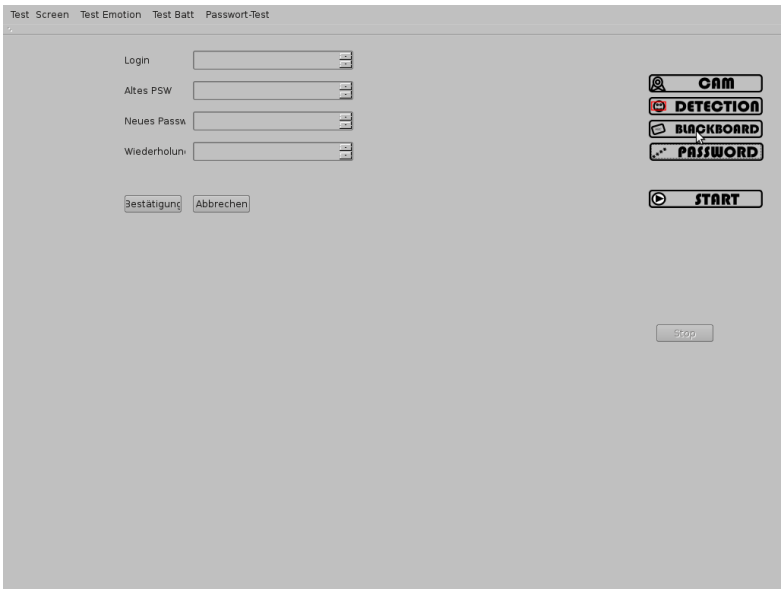


Abbildung B.11: Adminbildschirm: Passwortfunktionen

Abbildung B.12: Adminbildschirm: Personenerkennung

B.2 Evaluation der Bluetooth-Abstandsmessung

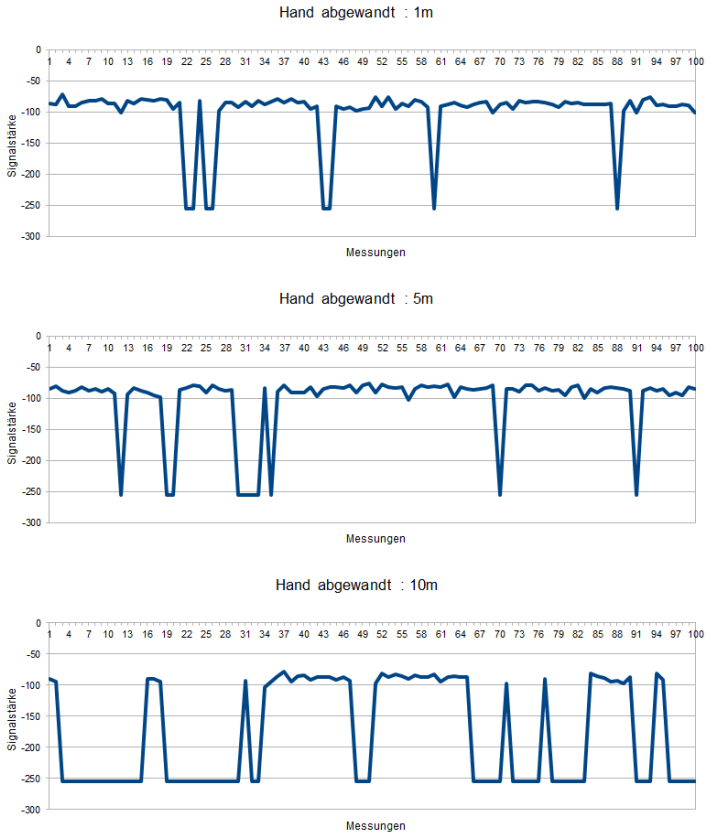


Abbildung B.13: Die Ergebnisse aus 100 Messungen. Der Nutzer hat das Handy in der Hand und ist dem Roboter stets abgewandt.

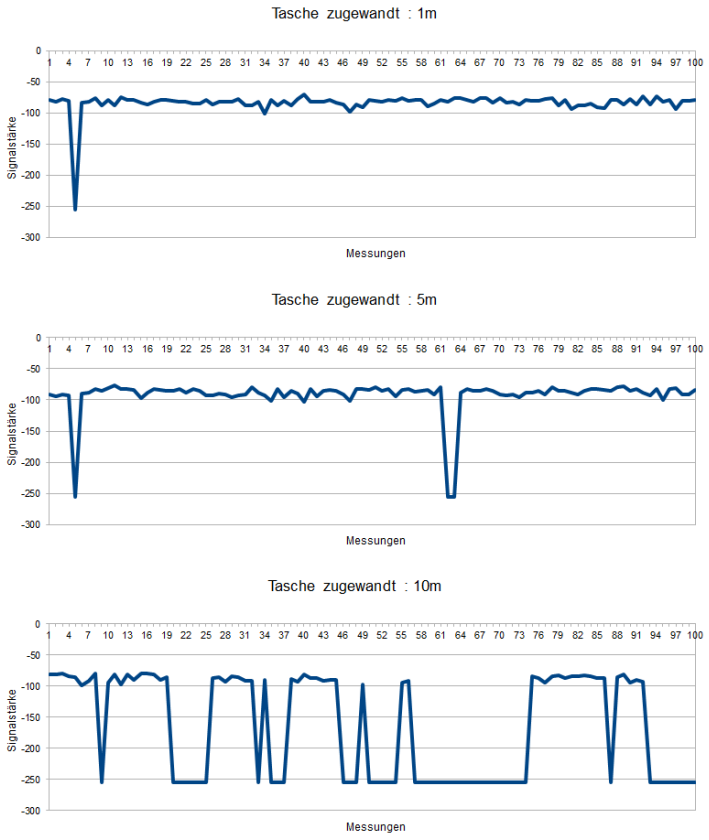


Abbildung B.14: Die Ergebnisse aus 100 Messungen. Der Nutzer hat das Handy in der Tasche und ist dem Roboter stets zugewandt.

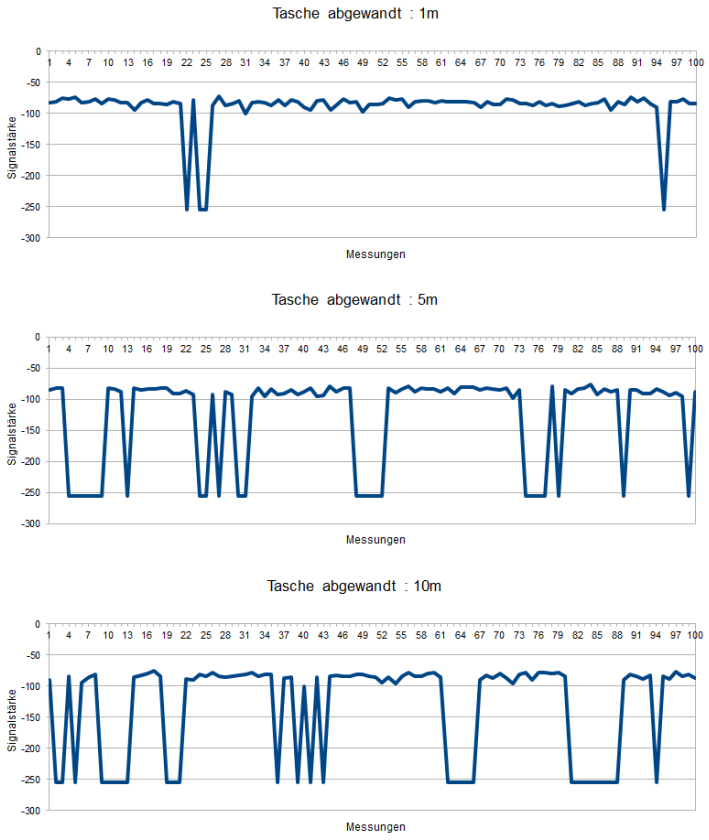


Abbildung B.15: Die Ergebnisse aus 100 Messungen. Der Nutzer hat das Handy in der Tasche und ist dem Roboter stets abgewandt.

B.3 Evaluation des Beinpaardetektors

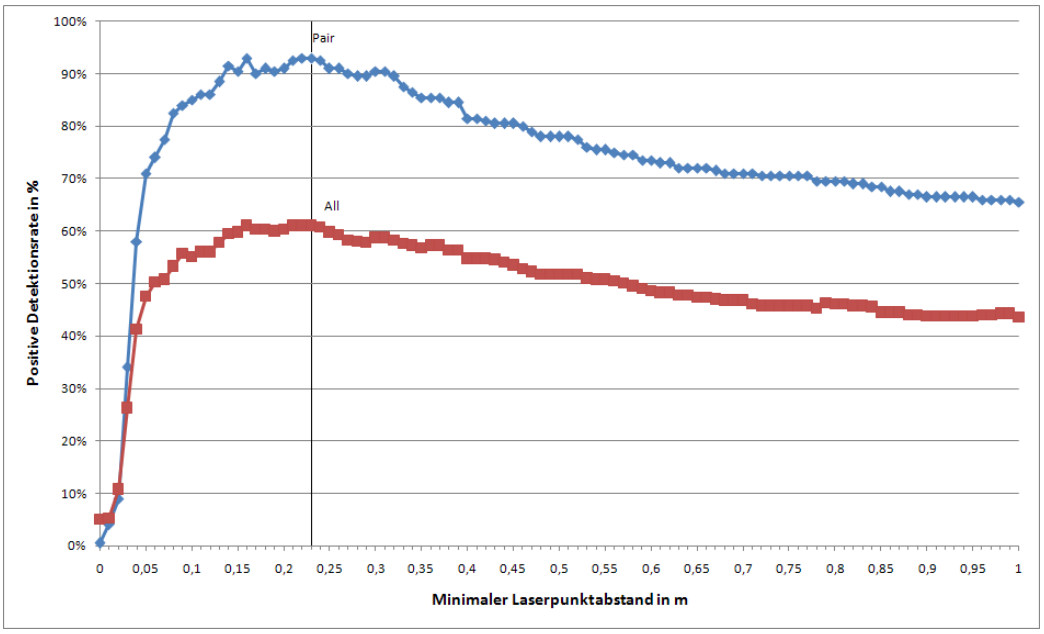


Abbildung B.16: Verlauf bei Veränderung des Parameters NEWOBJ. ◇ = Datensatz PAIR. □ = Datensatz ALL

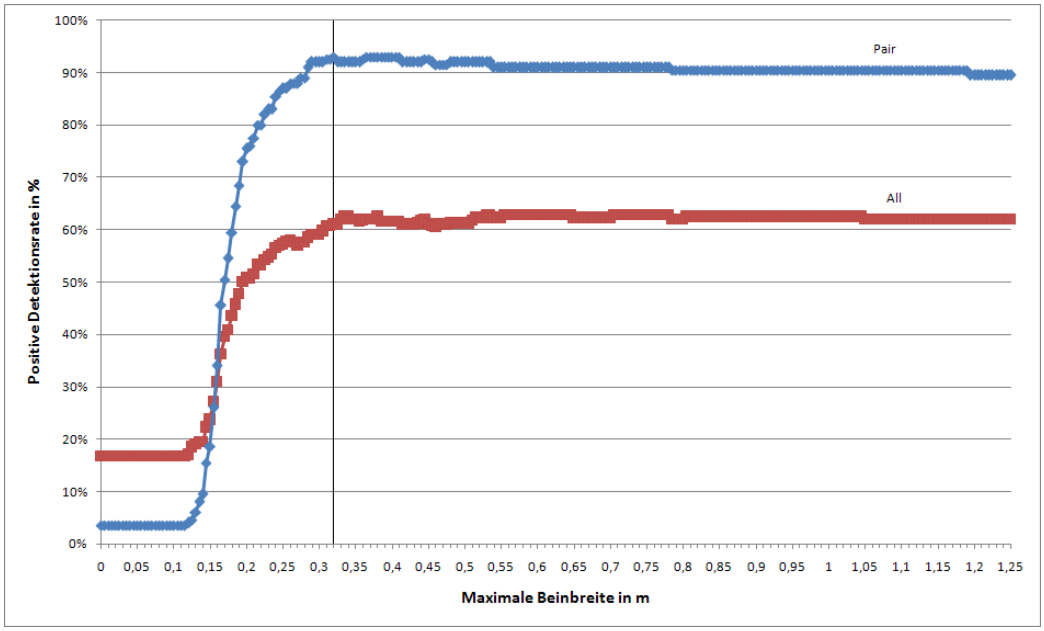


Abbildung B.17: Verlauf bei Veränderung des Parameters MAXW. ◇ = Datensatz PAIR. □ = Datensatz ALL

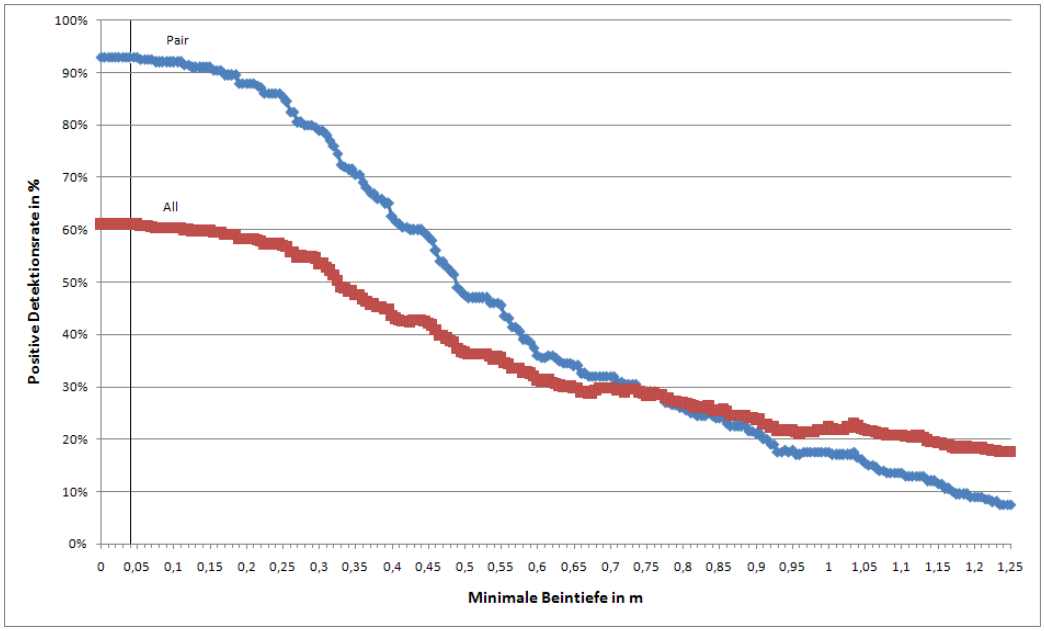


Abbildung B.18: Verlauf bei Veränderung des Parameters MIND. ◇ = Datensatz PAIR. □ = Datensatz ALL

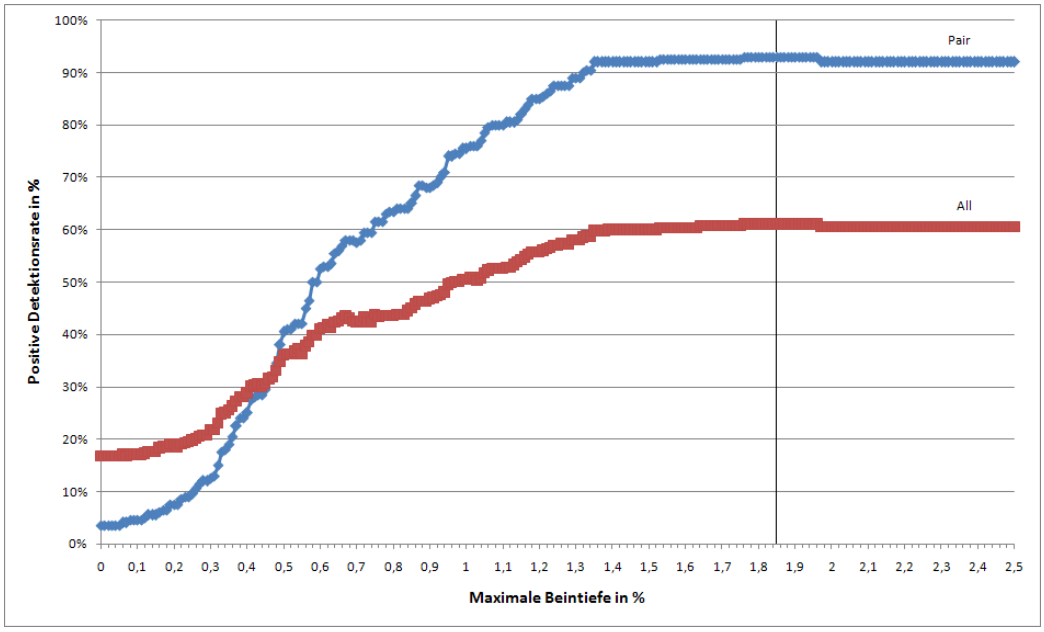


Abbildung B.19: Verlauf bei Veränderung des Parameters MAXD. ◇ = Datensatz PAIR. □ = Datensatz ALL

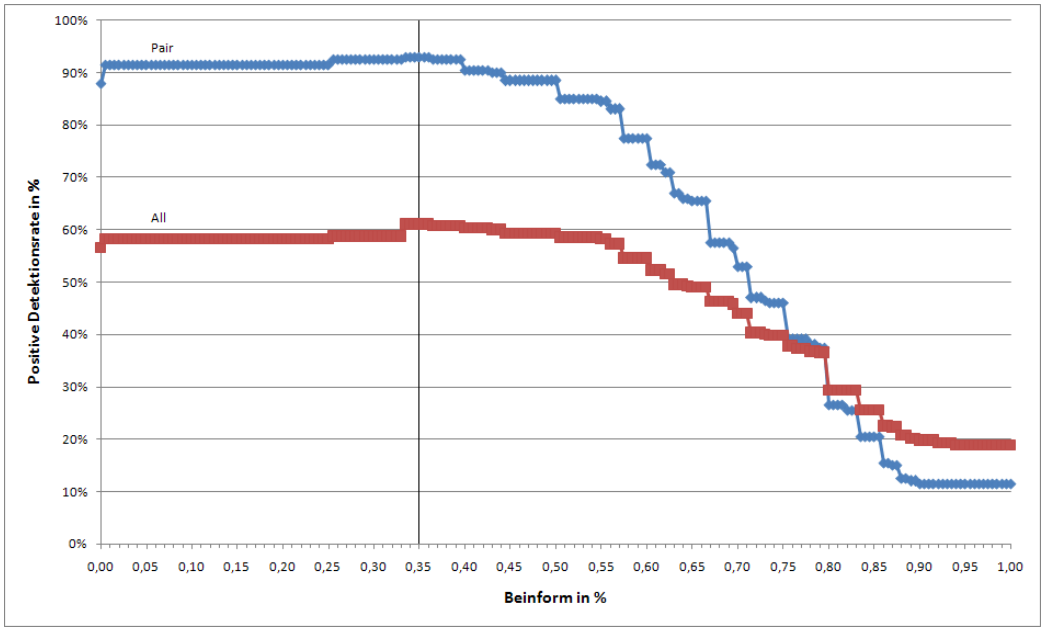


Abbildung B.20: Verlauf bei Veränderung des Parameters FORM. ◇ = Datensatz PAIR. □ = Datensatz ALL

B.4 Evaluation des Personendetektors

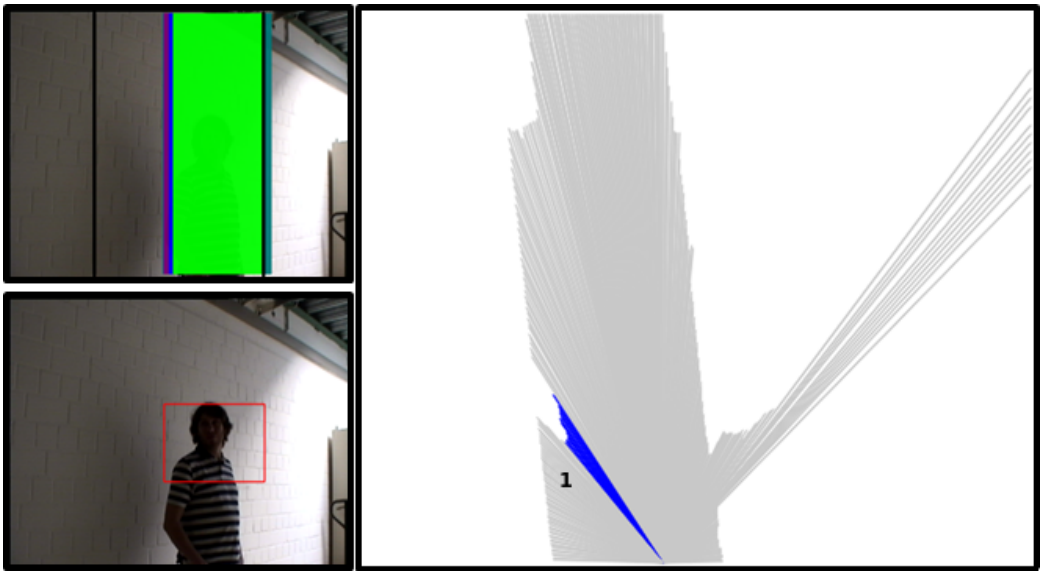


Abbildung B.21: Positives Beispiel 2. Die Person wurde trotz nicht idealer Beinposition detektiert und auf vier von fünf Bildern erkannt (1 = Rockobjekt (Person/Ziel))

.

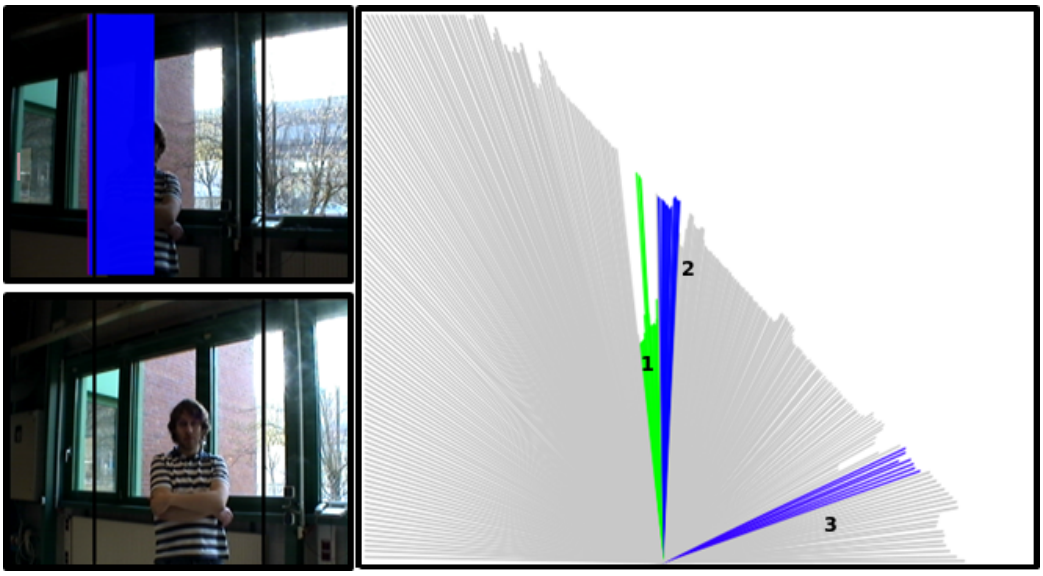


Abbildung B.22: Negatives Beispiel 2. Die Person wurde vom Gesichtsdetektor nicht erkannt. Darauf wurde die zweite Position in derselben Richtung untersucht und hier diesmal die Person erkannt, worauf der Roboter zum falschen Ziel fuhr (1 = Beinpaar (Person). 2 = Rockobjekt (Ziel). 3 = Rockobjekt)

.

Anhang C

Benutzerstudie

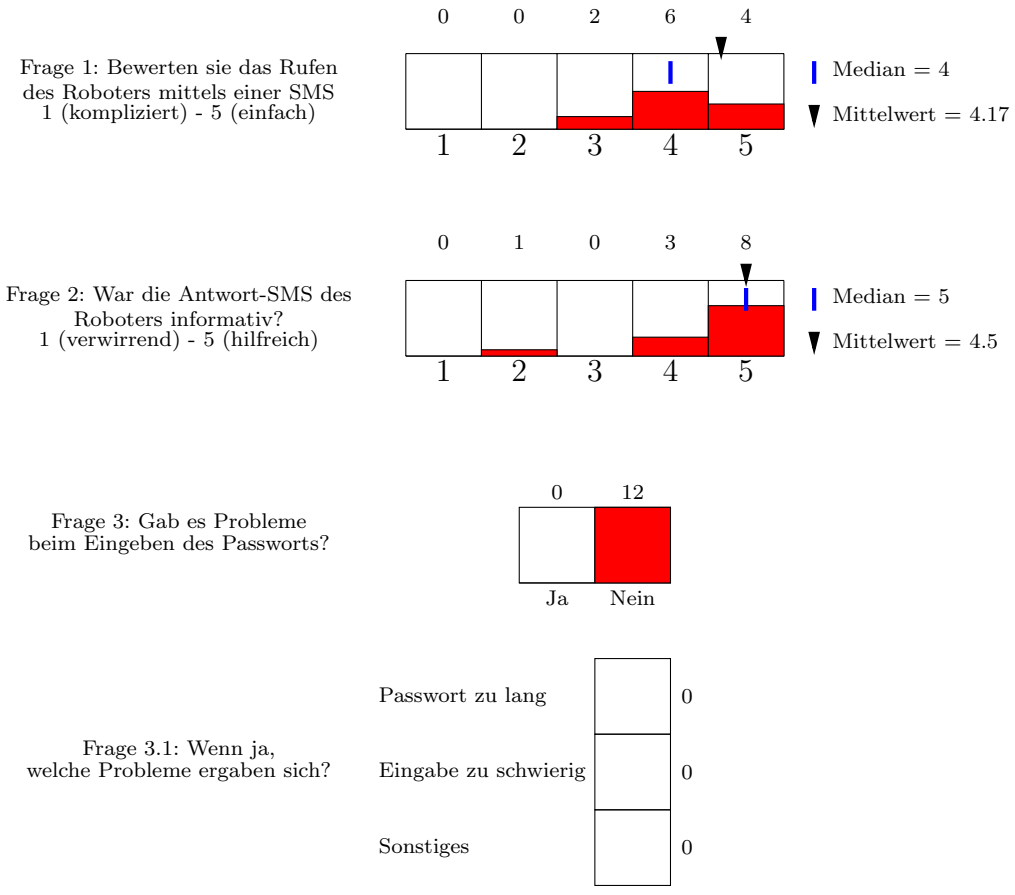
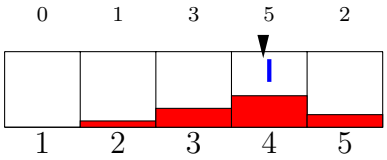


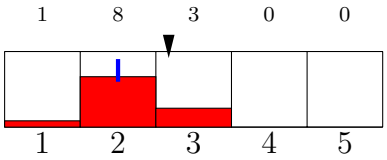
Abbildung C.1: Auswertung der Benutzerstudie

Frage 4: Bewerten Sie die Auswahl des Zielpunkts über den Touchscreen.
1 (verwirrend) - 5 (gut verständlich)



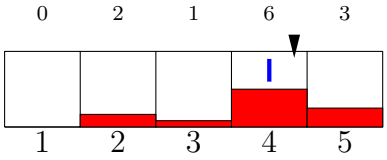
Median = 4
Mittelwert = 3.42

Frage 5: Bewerten Sie die Geschwindigkeit des Roboters.
1 (zu langsam) - 3 (zu schnell)



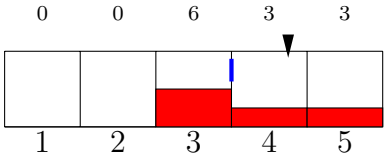
Median = 2
Mittelwert = 2.17

Frage 6: Bewerten sie die Hindernisvermeidung der Roboters.
1 (verwirrend) - 5 (verständlich)



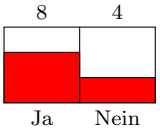
Median = 4
Mittelwert = 3.83

Frage 7: Hat das „Gesicht“ des Roboters die Bedienung erleichtert oder erschwert?
1 (erleichtert) - 5 (erschwert)



Median = 3.5
Mittelwert = 3.75

Frage 8: Haben Sie versucht den Pfad des Roboters zu blockieren um zu sehen wie er reagiert?



Frage 9: Haben Sie den Roboter, oder hat der Roboter sie während der Führung verloren?

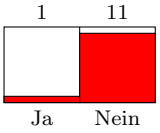
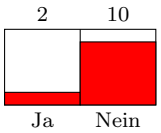
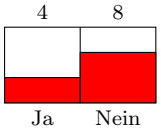


Abbildung C.2: Auswertung der Benutzerstudie

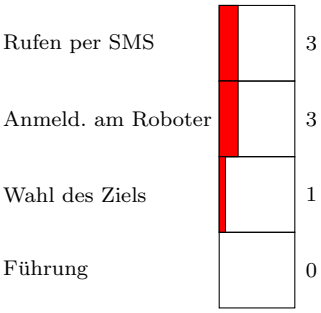
Frage 10: Gab es Pausen/Behinderungen/Wartezeiten während der Führung?



Frage 11: Mussten Sie einen Schritt der Bedienung mehrmals durchführen?



Frage 11.1: Bei welchem Schritt traten die Probleme auf?



Frage 12: Bewerten Sie die Benutzbarkeit von CampusGuide auf einer Skala von 1 (kompliziert) bis 5 (hilfreich).

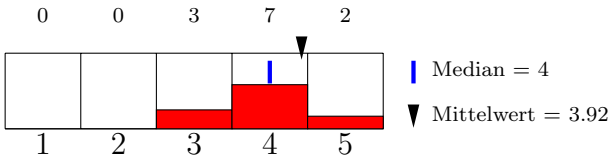


Abbildung C.3: Auswertung der Benutzerstudie

Projektgruppe CampusGuide

Herzlich willkommen beim Benutzertest von CampusGuide. Der Roboter soll die Aufgabe eines Fremdenführers für Gäste auf dem Gelände der TU Dortmund erfüllen. Dieser Test soll in kleinerem Maßstab im IRF statt finden. Orte die Sie als Ziel wählen können sind durch Schilder repräsentiert. Eine Liste aller Wegpunkte ist auf der Rückseite zu finden.
Bitte führen Sie die folgenden Schritte durch und befolgen Sie die Anweisungen, die Sie vom Roboter per SMS und per Touchscreen erhalten:

1. Rufen Sie CampusGuide mit einer SMS an die Nummer xxxx/xxxxxxx. Diese soll ihren momentanen Aufenthaltsort enthalten.
2. Warten Sie bis der Roboter an ihrer Position ankommt.
3. Melden Sie sich mit Ihrem Passwort am Roboter an.
4. Wählen Sie ein beliebiges Ziel aus der Liste und lassen Sie sich von CampusGuide dort hinführen.
5. Wählen Sie den Punkt "Neues Ziel Wählen" und lassen Sie sich zum Wegpunkt Ausgang führen.
6. Beenden Sie die Führung und schicken sie den Roboter nach Hause.
7. Füllen Sie den Evalutationsbogen aus.

Vielen Dank, dass Sie am Benutzertest von CampusGuide teilgenommen haben. Die von Ihnen gemachten Angaben werden anonym im Endbericht der Projektgruppe verwendet.

Abbildung C.4: Handout

Evaluationsbogen: CampusGuide

1	Zu Beginn des Tests haben Sie den Roboter per SMS gerufen. Empfanden Sie das Verfahren als zu kompliziert oder ging es leicht von der Hand? Bewerten Sie auf einer Skala von 1 - 5 (1: kompliziert – 5: einfach)	1 ☐ ☐ ☐ ☐ ☐ 5
2	War die SMS-Antwort des Roboters informativ oder blieben Fragen offen? (1: verwirrend – 5: hilfreich)	1 ☐ ☐ ☐ ☐ ☐ 5
3	Gab es Probleme beim Eingeben des Passworts?	ja (weiter mit 3.1) ☐ ☐ nein (weiter mit 4)
3.1	Welche Probleme ergaben sich? (Mehrfachnennung möglich)	☐ Passwort zu lang ☐ Eingabe ist zu schwierig ☐ Sonstiges: _____
4	Bewerten Sie die Auswahl des Zielpunkts über den Touchscreen. (1: verwirrend – 5: gut verständlich)	1 ☐ ☐ ☐ ☐ ☐ 5
5	Bewerten Sie die Geschwindigkeit des Roboters. (1: zu langsam, 3: genau richtig, 5: zu schnell)	1 ☐ ☐ ☐ ☐ ☐ 5
6	Der Roboter musste einem Hindernis ausweichen. Empfanden Sie die Reaktion des Roboters als nachvollziehbar oder verwirrend? (1: verwirrend – 5: verständlich)	1 ☐ ☐ ☐ ☐ ☐ 5
7	Hat das „Gesicht“ des Roboters die Bedienung erleichtert oder erschwert? (1: erschwert – 5: erleichtert)	1 ☐ ☐ ☐ ☐ ☐ 5
8	Haben Sie versucht den Pfad des Roboters zu blockieren um zu sehen wie er reagiert?	ja ☐ ☐ nein
9	Haben Sie den Roboter, oder hat der Roboter sie während der Führung verloren?	ja ☐ ☐ nein
10	Gab es Pausen/Behinderungen/Wartezeiten während der Führung?	ja ☐ ☐ nein
11	Mussten Sie einen Schritt der Bedienung mehrmals durchführen?	ja (weiter mit 11.1) ☐ ☐ nein (weiter mit 12)
11.1	Bei welchem Schritt traten die Probleme auf? (Mehrfachnennung möglich)	☐ Rufen per SMS ☐ Anmeldung am Roboter ☐ Wahl des Ziels ☐ Führung des Roboters
12	Bewerten Sie die Benutzbarkeit von CampusGuide auf einer Skala von 1 (nicht hilfreich, kompliziert) bis 5 (hilfreich, intuitiv verständlich).	1 ☐ ☐ ☐ ☐ ☐ 5

Abbildung C.5: Fragebogen

Literaturverzeichnis

[1] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, pp. 3–55, 1999.

[2] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1999–2005, 1999.

[3] G. Kim, W. Chung, K. Kim, M. Kim, S. Han, and R. Shinn, "The autonomous tour-guide robot jinny," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3450–3455, 2004.

[4] H. Gross, H. Böhme, C. Schröter, S. Müller, A. König, C. Martin, M. Merten, and A. Bley, "Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3471–3478, 2008.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Intelligent robotics and autonomous agents, MIT Press, 2005.

[6] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1311–1318, MIT Press, 2001.

[7] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 11–18, 1998.

[8] D. Fox, W. Burgard, S. Thrun, and A. Cremers, "Position estimation for mobile robots in dynamic environments," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 983–988, 1998.

[9] A. Puntambekar, "Terrain modeling and obstacle detection for unmanned autonomous ground robots," Master's thesis, The Center for Advanced Computer Studies, University of Louisiana at Lafayette, 2006.

[10] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens, and S. Strohband, "Winning the darpa grand challenge with an ai robot," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 17–20, 2006.

[11] U. Ozguner, K. A. Redmill, and A. Broggi, "Team terramax and the darpa grand challenge: a general overview," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 232–237, 2004.

[12] Y. Morales, E. Takeuchi, A. Carballo, W. Tokunaga, H. Kuniyoshi, A. Aburadani, A. Hirosawa, Y. Nagasaka, Y. Suzuki, and T. Tsubouchi, "1km autonomous robot navigation on outdoor pedestrian paths running the tsukuba challenge 2007," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 219–225, 2008.

[13] M. H. Bruch, G. A. Gilbreath, J. W. Muelhauser, and J. Q. Lum, "Accurate waypoint navigation using non-differential GPS," in *AUVSI Unmanned Systems, Lake Buena Vista, FL, July 9-11*, 2002.

[14] A. Bartel, F. Meyer, C. Sinke, T. Wiemann, A. Nüchter, K. Lingemann, and J. Hertzberg, "Real-time outdoor trail detection on a mobile robot," in *Proceedings of the IASTED International Conference on Robotics and Applications (RA)*, pp. 477–482, 2007.

[15] M. Blas, M. Agrawal, A. Sundaresan, and K. Konolige, "Fast color/texture segmentation for outdoor robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4078–4085, 2008.

[16] C. Bishop, *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

[17] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment," *Journal of Robotic Systems*, vol. 18, no. 10, pp. 577–587, 2001.

[18] S. Ioffe and D. A. Forsyth, "Probabilistic methods for finding people," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 45–68, 2001.

[19] J. Zhou and J. Hoang, "Real time robust human detection and tracking system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Workshops*, p. 149, 2005.

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, p. 511, 2001.

[21] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, pp. 153–161, 2005.

[22] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1491–1498, 2006.

[23] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 24–31, 2009.

[24] K. Dautenhahn, "Socially intelligent robots: dimensions of human–robot interaction," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, no. 1480, pp. 679–704, 2007.

[25] J. Schulte, C. Rosenberg, and S. Thrun, "Spontaneous, short-term interaction with mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 658–663, 1999.

[26] M. Merten and H.-M. Gross, "Highly adaptable hardware architecture for scientific and industrial mobile robots," in *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 1130–1135, 2008.

[27] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[28] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[29] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2432–2437, 2005.

[30] B. Dezső, A. Jüttner, and P. Kovács, "LEMON–an Open Source C++ Graph Template Library," in *Workshop on Generative Technologies 10*, pp. 3–13, 2010.

[31] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[32] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1572–1577, 1998.

[33] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, 1985.

[34] P. Prodanov, A. Drygajlo, G. Ramel, M. Meisser, and R. Siegwart, "Voice enabled interface for interactive tour-guide robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1332–1337, 2002.

[35] B. Jensen, N. Tomatis, L. Mayor, A. Drygajlo, and R. Siegwart, "Robots meet humans-interaction in public spaces," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 6, pp. 1530–1546, 2005.