Endbericht der Projektgruppe 525

PARTYBOT

Entwicklung einer Aufmerksamkeitsarchitektur für
einen mobilen Roboter auf der Basis von OSGi

Teilnehmer

Faruk Bagdat, Matthias Bahne, Ruxandra Cernat, Timo Häußler,
Alexander Kandelberg, Fabian Knobloch, Olaf Neugebauer, Christian Niegl,
Axel Plinge, Norman Schmitz, Torben Wiggerich, Tim Wilking

Betreuer

Prof. Dr.-Ing. Gernot A. Fink,
Dipl.-Ing. Jan Richarz

TU Dortmund
30. März 2009

Inhaltsverzeichnis

1 Einführung **7**
1.1 Roboter in häuslichen Umgebungen . . . 7
1.2 Zielsetzung und Voraussetzungen . . . 8
1.3 Konzeptionelle Überlegungen . . . 9
1.4 Grundlagen . . . 10
1.5 Aufbau des Endberichts . . . 12

2 Personendetektion **13**
2.1 Multisensorische Personendetektion auf mobilen Robotern . . . 13
2.1.1 Personendetektion . . . 14
2.1.2 Integrationsmodelle . . . 14
2.1.3 Verfahren . . . 16
2.2 Beinpaardetektion . . . 16
2.2.1 Stand der Technik . . . 16
2.2.2 Verfahren . . . 17
2.3 Gesichtsdetektion . . . 19
2.3.1 Stand der Technik . . . 19
2.3.2 Verfahren . . . 22
2.4 Ganzkörper- und Oberkörperdetektion . . . 24
2.4.1 Stand der Technik . . . 24
2.4.2 Verfahren . . . 25
2.5 Integration . . . 32
2.5.1 Kamerasteuerung . . . 32
2.5.2 Zusammenfassen der Detektionen . . . 33

3 Navigation in dynamischen Umgebungen **37**
3.1 Stand der Technik . . . 38
3.2 Dynamische Wegplanung . . . 39
3.2.1 Treiberimplementierung . . . 41
3.2.2 Integration in das Framework durch OSGi . . . 45
3.3 Reaktive Navigation . . . 46
3.3.1 Lasergestützte Personenverfolgung . . . 46
3.3.2 Angstverhalten . . . 46

4 Steuerarchitektur **47**
4.1 Steuerarchitekturen für mobile Roboter . . . 47
4.1.1 Reaktive Steuerarchitekturen . . . 48
4.1.2 Deliberative Steuerarchitekturen . . . 49
4.1.3 Hybride Steuerarchitekturen . . . 49
4.1.4 Verhaltensbasierte Steuerarchitekturen . . . 49
4.1.5 Resumee . . . 51
4.2 Aufmerksamkeitsbasierte Steuerarchitektur . . . 52
4.2.1 Use Cases . . . 52
4.2.2 Aufbau der Gesamtarchitektur . . . 53
4.2.3 Zustandsautomat . . . 53
4.3 Umsetzung in einer serviceorientierten Architektur . . . 54
4.3.1 OSGi . . . 54

4.3.2 SOA . . . 55
4.3.3 Funktionsweise des Frameworks . . . 55
4.4 Implementierung mit OSGi . . . 57
4.4.1 Programmierung . . . 58
4.4.2 Kommunikation über Sockets . . . 58
4.4.3 JNI . . . 58
4.4.4 Dienste . . . 59

5 Evaluierung 63
5.1 Informelle Tests . . . 63
5.2 Konfiguration und Evaluierung der HOG-Basierten Detektoren . . . 63
5.2.1 Vorgehensweise . . . 65
5.2.2 HOG Konfigurationen . . . 67
5.2.3 Klassifikatoren . . . 69
5.2.4 Suchraum . . . 71
5.2.5 Clustering . . . 72
5.2.6 Bewertung . . . 79
5.3 Evaluierung der Gesichtsdetektion . . . 81
5.3.1 Parameteroptimierung . . . 81
5.3.2 Kaskade . . . 82
5.3.3 Bewertung . . . 82
5.4 Evaluierung der kamerabasierten Detektoren im Vergleich . . . 83
5.4.1 Testszenarios . . . 83
5.4.2 Ergebnisse . . . 84
5.4.3 Bewertung . . . 84
5.5 Evaluierung des Beindetektors . . . 85
5.5.1 Szenarien . . . 85
5.5.2 Daten . . . 87
5.5.3 Bewertung . . . 90
5.6 Evaluierung der Detektoren mit Kamerabewegung . . . 90
5.6.1 Szenario . . . 91
5.6.2 Daten . . . 91
5.6.3 Bewertung . . . 93
5.7 Evaluierung des Clusterings . . . 93
5.7.1 Bewertung . . . 94
5.8 Versuche mit der dynamischen Wegplanung . . . 95
5.8.1 Szenario 1 . . . 96
5.8.2 Szenario 2 . . . 97
5.8.3 Schwierigkeiten bei der Evaluierung . . . 98
5.8.4 Bewertung . . . 98
5.9 Systemtests . . . 99
5.9.1 Szenario 1: Finden und Ansprechen einer Person aus einer Gruppe . . . 99
5.9.2 Szenario 2: Finden und Ansprechen einer alleinstehenden Person . . . 101
5.9.3 Szenario 3: Finden und Ansprechen einer entfernten Person . . . 101
5.9.4 Bewertung . . . 102

6 Fazit 103

A SCITOS Bedienung 113

B GUI **113**

C Parameter und Skripte **116**
C.1 Dynmap 116

D Werkzeuge und Hilfsprogramme **117**
D.1 Kamera 117
D.1.1 Kameraserver 117
D.1.2 ViscaThere 118
D.1.3 Recorder 118
D.2 Personendetektion 118
D.2.1 genhogtraindata 118
D.2.2 FANNTrainer 119
D.2.3 svm-train 119
D.2.4 PDValidate 119
D.2.5 HSDetect 119
D.2.6 PDDemo 119
D.2.7 FaceDetection 120
D.3 Player 120
D.4 Diverse 122
D.4.1 EBCTool 122

1 Einführung

Roboter haben schon seit vielen Jahrzehnten Einzug in die Industrie gehalten, wo sie Arbeiten erledigen, die Präzision, Geschwindigkeit und Wiederholungsgenauigkeit erfordern. Diese Art von Industrierobotern sind dem Menschen weit überlegen und sind mittlerweile aus der Produktion nicht mehr wegzudenken. Die Kosten für Hard- und Software, die für ein Robotersystem benötigt werden, sind heutzutage nicht mehr so kostenintensiv und die Anschaffung eines Roboters ist deswegen nicht mehr nur der Industrie vorbehalten. Die Roboterforschung beschäftigt sich schon seit vielen Jahren mit der Entwicklung von intelligenten, mobilen und autonomen Robotern. Solche Roboter haben einen vielseitigen Einsatzbereich. Inzwischen sind sie auch vermehrt als Serviceroboter in Haushalten wiederzufinden. Sie unterstützen den Menschen bei Reinigungsarbeiten in schwer zugänglichen Umgebungen, wie zum Beispiel der Fensterreinigungsroboter Raccoon von Procter & Gamble und dem Fraunhofer IPA [54], überwachen selbstständig Gebäude (siehe Kapitel 1.1), dienen als Helfer bei Inspektionen in der Rolle als Kanalroboter [22] oder unterstützen die Polizei und das Militär bei der medizinischen Verpflegung von verletzten Personen und bei der Entschärfung von Bomben [63].
Im Gegensatz zu Industrierobotern ist der Handlungs- und Bewegungsraum von mobilen Robotersystemen hochgradig dynamisch, da eine intelligente Interaktion mit Menschen und der Umwelt vorausgesetzt wird. Interessant sind vor allem solche Systeme, die sich in die Umgebung des Menschen eingliedern lassen und ebenfalls ein menschenähnliches Verhalten an den Tag legen. Dies erfordert allerdings weitaus komplexere Fähigkeiten als das reine Ausführen einer bestimmten Tätigkeit. Grundlegend für ein derartiges Verhalten ist das Handeln durch Eigenitiative, also Autonomie.

1.1 Roboter in häuslichen Umgebungen

Unter den mobilen Robotern gibt es zahlreiche Haushaltshelfer, wie Trilobite von Electrolux [54], der selbstständig den Fußboden saugt. Sein reaktives Verhalten wird in folgender Arbeitsweise ersichtlich: Trilobite sucht die Wand des Raumes, in dem er gestartet wurde, und fährt dann den Raum gegen den Uhrzeigersinn ab. Nach der Umkreisung fährt er zufällig weiter quer durch den Raum, ohne dabei zu planen. Mit Hilfe seiner Ultraschallsensoren versucht er den Hindernissen im Raum auszuweichen. Dazu ändert er seine Richtung zufällig, wenn er den Hindernissen näher kommt. Da der Ladestand des Akkus stets bekannt ist, kann der Roboter zur Aufladestation fahren, bevor der Akku komplett leer ist. Es gibt mehrere Situationen, in denen das Eingreifen des Menschen notwendig wird, beispielsweise beim Leeren des Staubsacks oder sobald der Roboter sich festgefahren hat. Deswegen ist dieser Roboter nicht als ganz autonom einzuordnen. Neben Trilobite existieren noch weitere Staubsaugerroboter auf dem Markt, wie zum Beispiel Roomba 560 von iRobot und RoboCleaner 3000 von Kärcher [27]. Sie haben alle ein reaktives Verhalten gemeinsam und sind nur auf primitive Arbeiten beschränkt.
Für weitaus komplexere Aufgaben als automatisches Staubsaugen sind Roboter wie der mobile Sicherheitsroboter Mosro [54, 53], RHINO der Universität Bonn [59], der mobile Shoppingassistent Toomas der TU Ilmenau [19], verschiedene Roboter des Fraunhofer IPA wie der Care-O-bot [20] oder der Partyfotografroboter Lewis [10] entwickelt worden. Diesen Robotern stehen mehrere und verschiedene Arten von Sensoren zur Verfügung, mit denen sie ihre Umwelt wahrnehmen können.
Der Wachroboter Mosro, der dynamische Umgebungen bewachen soll, ist zum Beispiel zusätzlich mit

Bewegungs-, Rauch- und Geruchssensoren ausgestattet und verfügt über Mikrofone und Kameras. Mit Hilfe dieser Sensoren und Aktoren ist es ihm möglich, selbstständig durch Gebäude zu navigieren.
Ein Forschungsteam der Universität Ilmenau hat den mobilen Shoppingassistenten Toomas entwickelt. Er soll den Kunden bei der Artikelsuche völlig autonom unterstützen. Dazu führt der Roboter die Kunden zu den gewünschten Artikeln. Toomas basiert auf der SCITOS A5 Plattform und ist bestückt mit einer Kamera, die dem Roboter eine Rundumsicht von 360° ermöglicht.
Des Weiteren existieren zahlreiche Forschungsergebnisse vom Fraunhofer IPA. Sehr interessant ist der Roboter Care-O-bot [20], der mittlerweile in der dritten Generation existiert. Dieser Serviceroboter soll im Haushalt schwere, monotone oder lästige Arbeiten für den Menschen übernehmen. Er kann typische Haushaltgegenstände wie Tassen und Gläser erkennen und Getränke servieren. Der Roboter verfügt über einen hochflexiblen Arm mit sieben Freiheitsgraden und ist ausgestattet mit Stereovision-Farbkameras, Laserscannern und einer 3-D-Tiefenbildkamera. Damit erfasst der Roboter seine Umgebung in Echtzeit und 3-D und kann anhand der Daten auch in dynamischen Umgebungen sicher manövrieren, Objekte erkennen, lokalisieren und handhaben.
Das IPA hat ebenso drei Roboter entwickelt, die zusammen als Team im Museum für Kommunikation in Berlin als Begrüssungskomitee arbeiten [18]. Das Besondere an den Robotern ist, dass jeder einzelne eine andere Aufgabe ausführt. Der Erste wurde „Komm-Rein“ getauft und fährt in einem speziellen Besuchermodus. Wie sein Name verrät, ist er für die Begrüßung zuständig. Bereits begrüßte Personen werden gespeichert und dadurch nicht ein weiteres Mal begrüßt. Der Zweite heißt „Also-Gut“ und erklärt den Besuchern im Programmmodus die Ausstellungsstücke des Museums. Synchron zu den abgespielten Multimediadateien kann er seinen Kopf bewegen. Der Roboter namens „Mach-Was“ vervollständigt das Team aus drei Robotern. Dieser spielt im Ballmodus mit einem großen Ball. Dabei ändert er ständig seine Richtung, um bei den Besuchern kindlich und verspielt zu wirken. Die Roboter können miteinander kommunizieren. Das geschieht über Funk-Ethernet und eine Zentrale, die die Signale entgegen nimmt und koordiniert.
Der Partyfotograf Lewis, der an der Washington University entwickelt wurde, bewegt sich frei durch den Raum und detektiert Menschen anhand ihrer Hautfarbe. Seine Hauptaufgabe ist es, die Personen zu fotografieren, die er auf seinen Fahrten durch die Räume findet. Dabei vermeidet er Kollisionen mithilfe seiner Laser- und Stoßsensoren. Sein internes Odometer ermittelt dabei ständig die Position. Der Roboter ist komplett ereignisgesteuert. Ihm steht neben der Kamera zur Personendetektion noch ein fest installierter Fotoapparat zur Verfügung, mit dem die Fotos geschossen werden. Der Roboter besitzt darüber hinaus einen Lasersensor, der die benötigten Daten für die Pfadplanung und Hindernisvermeidung liefert.
Diese Art von Robotern sind weitaus interessanter als rein reaktive Systeme. Durch die Fähigkeit zur Mensch-Maschine-Interaktion entsteht ein lebendiges Verhalten, was auf den Menschen sehr attraktiv wirkt.

1.2 Zielsetzung und Voraussetzungen

Das Ziel der Projektgruppe ist es, einen PARTYBOT zu entwickeln, der auf diversen Veranstaltungen, wie zum Beispiel Partys, Cocktailabende, Informationsveranstaltungen etc., als Gast auftritt und mit seinem geselligen Verhalten als potentieller Interaktionspartner wahrgenommen wird. Es soll ein Robotersystem entstehen, das sich auf einer Veranstaltung mit vielen Menschen frei bewegt und sich stets zu interessanten Bereichen begibt. Interessant sind jene Bereiche, die auch für andere Gäste interessant sein können

und in denen sich folglich potentielle Interaktionspartner aufhalten. Insbesondere soll der PARTYBOT also selbstständig solche Bereiche detektieren, Personen finden und Kontakt mit ihnen aufnehmen. Ausschlaggebend für seinen Auftritt ist, bei der Kontaktaufnahme einzeln stehende Gäste zu bevorzugen oder sich zu Gruppen zu stellen, ohne das Gespräch zu stören. Bei erfolgreicher Kontaktaufnahme mit einem Gast kann er verschiedene Aktionen ausführen. Denkbar ist der Fall, dass die kontaktierte Person zusammen mit dem PARTYBOT zum Beispiel ins Foyer gehen möchte. Daher ist es wünschenswert, dass der Roboter einer Person hinterher fährt, wenn er über den integrierten Touchscreen dazu aufgefordert wird. Neben der Eigenschaft Interesse zu zeigen, soll der Roboter auch zum Selbstschutz Gefahren erkennen und dementsprechend reagieren können. Ein derartiges Furchtverhalten kann beispielsweise durch sehr laute Geräusche oder hastige Bewegungen anwesender Personen ausgelöst werden.

Dazu soll auf der vorhandenen mobilen Plattform SCITOS G5 eine serviceorientierte Kontrollarchitektur auf der Basis von OSGi realisiert werden. OSGi ist ein Softwareframework und bietet den Entwicklern einer serviceorientierten Architektur entscheidende Vorteile, auf die in Kapitel 4.3.1 näher eingegangen wird. Die Ansteuerung des Roboters erfolgt über das Open Source Programmpaket Player, welches den Zugriff auf die Sensoren und Aktoren bereitstellt und in das OSGi-Framework integriert werden muss. Eine zentrale Aufgabe ist die Identifizierung und Implementierung benötigter Services, wie zum Beispiel Hindernisvermeidung, Nutzerdialog und Personendetektion als OSGi-Bundles. Als Beispiel sei die Kollisionsvermeidung genannt, die einen Zusammenstoß des PARTYBOTS mit anderen Gästen und Gegenständen verhindern soll. Dabei werden anhand der Laser- und Sonarwerte Hindernisse erkannt und umfahren, und im Falle einer Kollisionsgefahr der Roboter über die Motorsteuerung angehalten. Die Pfadplanung zu einem bestimmten Punkt ist ein Beispiel für einen Service, welcher die Hindernisvermeidung nutzt. Insbesondere ist dafür die Kommunikation sowohl zwischen den OSGi-Bundles selbst, als auch zwischen den OSGi-Bundles und Player notwendig. Des Weiteren ist die Konzeption eines Aufmerksamkeitsmechanismus notwendig, der zur Erkennung von interessanten Bereichen, Personen oder Verhalten dient, um das erwünschte gesellige Verhalten zu verwirklichen. Grundlegend dafür ist eine robuste und multimodale Personendetektion, um potentielle Interaktionspartner detektieren zu können. Zur Verifizierung des Gesamtsystems ist der Entwurf diverser reaktiver Verhaltensmuster, wie zum Beispiel Interesse- bzw. Furcht-Verhalten notwendig.

1.3 Konzeptionelle Überlegungen

Für die erfolgreiche Umsetzung der erläuterten Zielsetzung wurden drei Hauptthemenbereiche bearbeitet. Diese sind Personendetektion, Navigation in dynamischen Umgebungen und Steuerarchitektur. Abbildung 1 illustriert das Zusammenwirken dieser Themenbereiche.

Das anvisierte Zielsystem erfordert als wichtigste Fähigkeit das Detektieren von Personen, da das Einsatzszenario für den PARTYBOT darin besteht, auf einer Veranstaltung mit vielen Personen in geschlossenen Räumlichkeiten sich frei zu bewegen und mit den Gästen zu interagieren. Damit der PARTYBOT als ein Gast wie jeder andere wahrgenommen wird, ist es wichtig, dass er ein menschenähnliches Verhalten an den Tag legt. Das Verwechseln von Menschen und Gegenständen, wie zum Beispiel Stühlen, Säulen, Tischen etc., sollte auf ein Minimum reduziert werden. Bei zu häufigen Fehldetektionen kann die Interaktion zwischen dem PARTYBOT und den Gästen scheitern. Somit ist eine robuste Personendetektion der maßgebliche Grundbaustein für die Folgeaktionen.

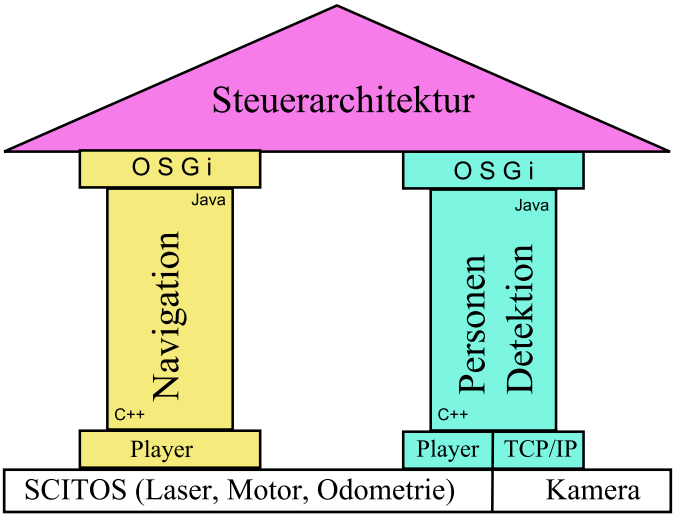


Abbildung 1: Konzeptionelle Aufteilung der Themenbereiche

Für eine erfolgreiche Interaktion muss sich der PARTYBOT zu erkannten potentiellen Nutzern hinbewegen können. Dies erfordert die Fähigkeit, beliebige Punkte im Raum erreichen zu können. Der Roboter befindet sich stets in einer hochgradig dynamischen Umgebung. So ist es unerlässlich, dass dynamische Hindernisse frühzeitig erkannt und umfahren werden. Dynamische Hindernisse können auch dazu führen, dass ein zu einem Zeitpunkt t geplanter Pfad zum Zeitpunkt $t + 1$ nicht mehr gültig ist, da ein Hindernis den Pfad unumgänglich blockiert. Mit einer dynamischen Pfadplanung kann in geeigneter Weise darauf reagiert und ein alternativer Pfad geplant werden.
Mit der Personendetektion und der Navigation ist der PARTYBOT in der Lage, Personen zu detektieren und zu ihnen hinzufahren. Allerdings ist eine Instanz erforderlich, die das Gesamtverhalten koordiniert und Entscheidungen trifft. Die aufmerksamkeitsbasierte Steuerarchitektur auf der Basis von OSGi kapselt alle Funktionalitäten des Roboters in OSGi-Bundles und bietet sie als Services an, so dass andere Bundles sie nutzen können. Zudem definiert sie verschiedene Verhaltensweisen und dient dazu, die Reaktionen auf die in seiner Umwelt auftretenden Ereignisse auszuwählen und zu koordinieren.

1.4 Grundlagen

Als Hardware kommt die mobile Roboterplattform SCITOS G5 der Firma MetraLabs GmbH zum Einsatz (siehe Abbildung 2). Der Roboter ist entwickelt worden um Forschungstätigkeiten in den Bereichen Mapping, Lokalisierung, Pfadplanung oder Mensch-Maschine-Interaktion zu unterstützen. Für diesen Zweck ist der Roboter mit einer ganzen Reihe Erweiterungsmodulen wie z.B. einer Kamera, einem Lasersensor, einem Sonarring und einem Touchscreen mit integrierten Lautsprechern ausgestattet. Des Weiteren steht zum Nachhalten der aktuellen Position die Odometrie zur Verfügung. Dabei verbindet die SCITOS G5 Plattform die Zuverlässigkeit von Industriestandards und die Flexibilität eines eingebetteten Systems. Gesteuert wird die Roboterplattform über einen eingebetteten PC, auf dem eine Fedora Core 6 Linux Distribution installiert ist. Die Energieversorgung wird mit einer wiederaufladbaren Batterie realisiert, mit welcher der Roboter bis

zu 12 Stunden ohne weitere Stromversorgung mobil sein kann. Die Kommunikation zwischen den einzelnen Modulen des Roboters läuft über einen internen CAN Bus.

Hardwarekomponenten im Detail

Der PC ist mit einem Intel Core Duo T2300 1,6 Ghz Prozessor und 2048 Mb Arbeitsspeicher ausgestattet und verfügt unter anderem über WLAN-Funktionalität. Der Laserdetektor hat eine Blickfeld von 270°. Durch die Aufbauten des Roboters ist der Bereich auf 180° eingeschränkt (siehe Abbildung 2 unten). Der Laser hat eine Auflösung von 0,5° und ist in einer Höhe von 40 cm angebracht und in Fahrtrichtung des Roboters orientiert. Die analoge Kamera des SCITOS G5 liefert Bilder in PAL Auflösung. Sie ist mit Motoren ausgestattet und mit diesen von -170 bis +170° schwenkbar und von -30 bis +90° neigbar. Über die Odometrie kann die aktuelle Position des Roboters in kartesischen Koordinaten und dessen Ausrichtung im Raum als Winkel ausgelesen werden. Die X- und Y-Position und der Winkel werden dabei inkrementell von einer Startposition bis zur aktuellen Position errechnet. Dazu hat jedes Rad einen Sensor mit einer Auflösung von 460 Schritten pro Radumdrehung. Als Antrieb kommen zwei Drei-Phasen-EC-Motoren zum Einsatz, welche jeweils eins von zwei zur Verfügung stehenden Rädern antreiben. Der Roboter hat eine Höchstgeschwindigkeit von 1,1 m/s und eine mögliche Drehgeschwindigkeit von 200°/s. Der Roboter verfügt über eine Zugkraft von 15 Kg und kann Steigungen bis zu 10° bewältigen

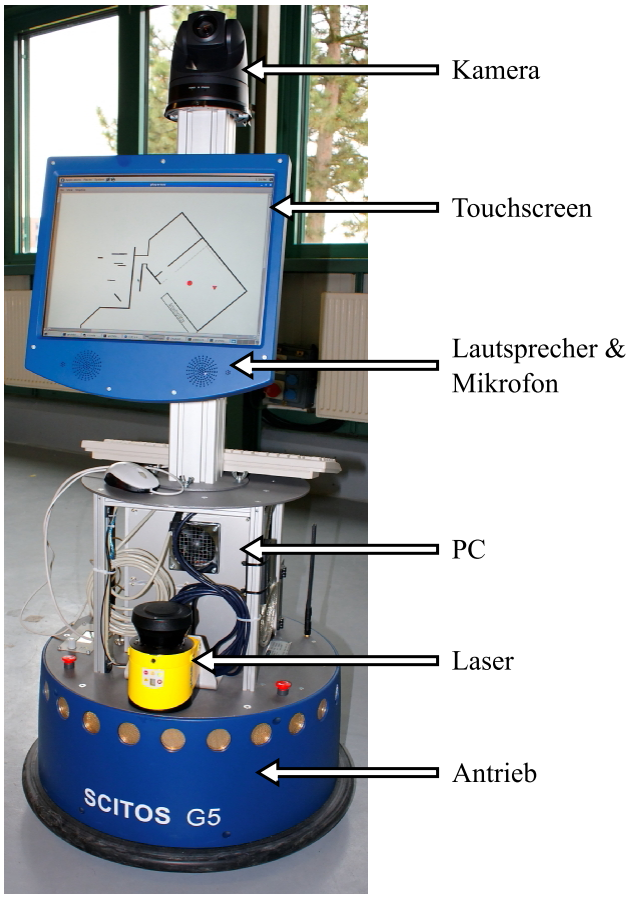


Abbildung 2: Der SCITOS G5 mit Kamera

1.5 Aufbau des Endberichts

Dieser Bericht beschreibt die Arbeit der Projektgruppe Partybot. Die Gliederung richtet sich nach den Themenbereichen, die sich im Rahmen der konzeptionellen Überlegungen (siehe Kapitel 1.3) herauskristallisiert haben. In Kapitel 2 werden diverse Verfahren vorgestellt, die der PARTYBOT für die Personendetektion benutzt. Im Vordergrund stehen dabei die verschiedenen Detektionsarten Gesichtsdetektion, Beinpaardetektion und Ober- und Ganzkörperdetektion, die kombiniert für die robuste Personendetektion sorgen. Die eingesetzten Verfahren, die auf Kamerabildern und Laserdaten basieren, werden vorgestellt. Auf die Navigation des Roboters wird in Kapitel 3 näher eingegangen. Es wird eine Weiterentwicklung des Wavefront-Navigationsalgorithmus beschrieben, die die Fortbewegung des Roboters in dynamischen Umgebungen verbessert. Das Kapitel 4 beschreibt die serviceorientierte Steuerarchitektur auf Basis von OSGi, die für die Steuerung des Gesamtverhaltens des PARTYBOT verantwortlich ist. Die Architektur basiert auf einem ereignisgesteuerten Zustandsautomaten. Anschließend werden die Ergebnisse der Evaluierung der Teilsysteme und des Gesamtsystems in Kapitel 5 beschrieben. Der Bericht schließt mit einem Fazit im Kapitel 6 ab.

2 Personendetektion

Der PARTYBOT hat ein spannendes und abwechslungsreiches Einsatzgebiet. Während andere Roboter in Fabrikhallen oder der Kanalisation vor sich hin arbeiten [22], den Menschen lästige Aufgaben wie Haushaltsreinigung abnehmen [27] oder in staubigen Museen herumstehen [18], darf der PARTYBOT am geselligen Treiben auf Partys teilhaben. Wie jeder gute Gastgeber weiß, sind die Gäste der wesentliche Bestandteil einer jeden Party. Damit der PARTYBOT zum Teil der Party wird, kann er Personen finden, ansprechen und verfolgen. Die Navigation wird im nächsten Kapitel erläutert. Hier beschäftigen wir uns zunächst einmal damit, wie Personen an einer bestimmten Raumposition detektiert werden.

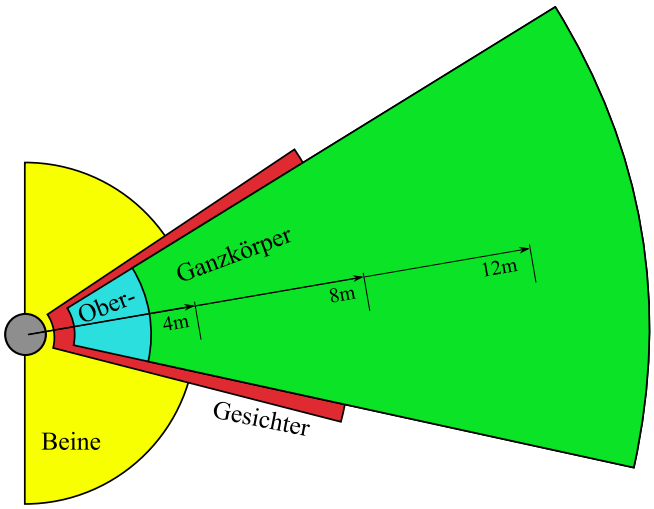


Abbildung 3: Entfernungsarbeitsbereich der Detektoren (Sicht von oben)

Der PARTYBOT setzt vier Detektoren ein, um Menschen in seinem Umfeld zu bemerken. Abbildung 3 zeigt deren Sichtfelder. Der Laser wird zur Detektion von Beinpaaren eingesetzt. Die dreh- und schwenkbare Kamera wird mit den sehr robusten HOG-Detektoren zur Ganz- und Oberkörperdetektion verwendet. Da der PARTYBOT sicherlich etliche Blicke auf sich zieht, werden auch zugewandte Gesichter detektiert.
Der PARTYBOT nutzt die Detektoren kombiniert in verschiedenen Verhaltensweisen, um ein aktuelles Bild seiner Umgebung zu erhalten. Dabei richtet er die Kamera auf vom Laser detektierte Objekte und sieht sich wiederholt um, nachdem er seine Position geändert hat. Dazu wird bei Annäherung an eine Person ihre Position in regelmäßigen Abständen neu überprüft.

2.1 Multisensorische Personendetektion auf mobilen Robotern

Viele mobile Roboter, die auf die eine oder andere Weise mit Menschen in Interaktion treten, verwenden verschiedene Sensoren, um Personen zu finden und zu verfolgen. Dabei werden insbesondere optische Sensoren oft von mehreren Detektoren genutzt. Demgegenüber werden Laser, Sonar und Mikrofonarrays in der Regel nur von einem Lokalisationsalgorithmus verwendet. Im folgenden Abschnitt wird eine kurze Übersicht über gebräuchliche Verfahren gegeben. Die Art der Kombination auf verschiedenen Modalitäten agierender Detektoren ist ein Charakteristikum der verwendeten integrierten Architektur. Der übernächste

Abschnitt stellt einige ausgewählte Ansätze vor. Im Anschluss wird der Ansatz beschrieben, der von uns auf dem PARTYBOT verwendet wird.

2.1.1 Personendetektion

Es existieren eine Vielzahl verschiedener Ansätze zur multisensorischen Personendetektion. Der Einsatz auf einem mobilen Roboter bringt einige zusätzliche Anforderungen mit sich, da die Umgebung weitgehend unbekannt ist und die Sensoren sich teilweise in Bewegung befinden. So sind bildbasierte Methoden für die Interaktion in einer festen Umgebung, die Differenzbilder und Hintergrundsubtraktion verwenden, nicht ohne eine adaptive Erweiterung verwendbar, welche die Bewegung der Kamera kompensiert bzw. das Hintergundmodell anpasst [16]. Demgegenüber sind Detektoren, welche in Einzelbildern Objekte finden, ohne Modifikation einsetzbar. Ein typisches optisches Merkmal für eine Person ist ihr Umriss oder vielmehr ihr Gradientenbild. Dies wird vielfach verwendet, um Personen in Bildern zu identifizieren [13, 44]. Hier werden oft Differenzbilder verwendet, weil durch Bewegung der Person oder durch Parallaxe bei der Eigenbewegung der Kamera, Objekte im Raum gut separiert werden [21, 16]. Auf Einzelbildern lassen sich auch Gesichtsdetektoren gut einsetzen, da zumindest beim Zeitpunkt der Kontaktaufnahme mit dem Roboter von einem zugewandten Gesicht ausgegangen werden kann [28, 40]. Darüber hinaus bilden Gesichter markante, konturenreiche Objekte und es stehen heute viele erprobte und schnelle Algorithmen zur Verfügung [65]. Auf vielen mobilen Robotersystemen werden Laserscanner zur Detektion von Beinpaaren eingesetzt [29, 35]. Zum Auffinden von Personen werden auch Mikrofonpaare zur auditiven Lokalisation eingesetzt, da sich aus dem Kreuzleistungsspektrum der Mikrofonsignale leicht eine Richtungsschätzung berechnen lässt [28]. Mit der kontinuierlich steigenden Rechenleistung werden immer öfter auch komplexere Methoden der Sprecherlokalisation verwendet [51].

2.1.2 Integrationsmodelle

Die auf mobilen Robotern zur Personendetektion implementierten Architekturen lassen sich grob in zwei Gruppen einteilen. Die erste integriert mehrere Sensoren in einer gemeinsamen Umgebungskarte, welche für weitere Auswertungen und für die Sensorsteuerung benötigt werden. In der zweiten Gruppe werden isolierte Detektoren auf einzelnen Modalitäten angewendet und nachträglich in einem Weltmodell verrechnet. Hier werden oft erprobte und verbreitete Verfahren der Mustererkennung verwendet und dann symbolisch oder probabilistisch kombiniert.

Frühe Sensorintegration

Ein aus der Neurologie stammendes Konzept ist das einer (multi)sensorischen Interessantheitskarte (englisch *saliency map*). Nach der sogenannten *feature-integration theory* [60] werden im posterior parietalen Kortex interessante Ereignisse in einer ortsbasierten Karte verortet. Davon ausgehend wurde eine Implementierung für Bilder entwickelt, welche herausstechende Bildbereiche detektiert [23]. Der jeweils interessanteste Ort wird in einer inhibierenden Karte eingetragen, so dass in der Folge der jeweils nächstinteressante Ort gefunden wird. Dieses Vorgehen imitiert den aus der Kognitionspsychologie bekannten Effekt der IOR (englisch *inhibition of return*) bei Blickbewegungen. Der Ansatz wird heute oft um weitere Modalitäten, wie z.B. Bewegungsmerkmale in Bildfolgen oder akustische Ortung, erweitert [51].

Der Roboter ICUB [51] ist ein interessantes Beispiel für eine solche Integration. Hier werden akustische und visuelle Ereignisse in Kugelkoordinaten in eine „ego-sphere“ genannte Interessantheitskarte eingetragen, welche den Kopf des humanoiden Roboters umspannt. Die visuelle Interessantheit wird in Skalenpyramiden der üblichen Merkmale Intensität, Farbwert (englisch *hue*), Richtung und einem aus Differenzbildern gewonnen Bewegungsmerkmal berechnet. Die akustische Interessantheit wird aus lokalisierten Tonquellen gebildet. Der ICUB besitzt als Nachbildung der menschlichen Physiologie zwei Mikrofone in je einem künstlichen spiralförmigen Außenohr, sodass der Drehungswinkel aus dem Zeitversatz (englisch *interaural time difference*, kurz ITD) und der Neigungswinkel aus der spektralen Verschiebung (englisch *interaural spectral difference*) berechnet werden kann.

Eine besondere Form der Erweiterung des Konzeptes der Interessantheitskarte stellt die Repräsentation der Gesichtsähnlichkeit dar [4]. Das Markieren von Bildbereichen, die einem Gesicht ähnlich sehen, kann als Vorverarbeitungsschritt für Gesichtsdetektion verwendet werden. Hier kann mit Kanten- und Farbmerkmalen gearbeitet werden.

Eine Reihe von Forschern ergänzen die merkmalsgetriebene Interessantheitskarte um eine zweite, aufgabengesteuerte Interessantheitskarte. Im einfachsten Fall werden hier schlicht aufgabenspezifische Bereiche markiert und die Ähnlichkeit der Merkmale mit denen des markierten Bereichs als zweite Maske verwendet, wie etwa beim VOCUS System. Dieser Mechanismus kann beispielsweise von einem mobilen Roboter genutzt werden, um selbsttätig eine Karte seiner Umgebung aufzubauen [17].

Kombination isolierter Detektoren

Viele Autoren verwenden anderweitig entwickelte isolierte Detektoren und kombinieren deren Ausgaben zu Personenhypothesen. Dies kann auf probabilistischer, numerischer oder symbolischer Ebene geschehen.

Beim HORUS Roboter der TU Ilmenau [35] wird eine statistische Verrechnung von Sensordaten vorgenommen. Entfernungsdaten von einem Laserscanner und Sonar werden mit farbbasierten Detektionen auf den Bildern einer Fischaugen-Rundumkamera kombiniert. Dabei werden die jeweiligen Ungenauigkeiten der Entfernungsschätzung der einzelnen Sensoren, sowie die Wahrscheinlichkeit, ob es sich bei dem detektierten Objekt um eine Person handelt, als zweidimensionale Gaußverteilung in einer roboterzentrierten Umgebungskarte modelliert. Die einzelnen Gaußverteilungen werden mittels eines flexiblen Schemas, der sogenannten Kovarianzintersektion (englisch *covariance intersection*, kurz CI) zusammengefasst. Dabei werden die sensorbasierten Personenhypothesen als Teil eines Markoffprozesses erster Ordnung aufgefasst. So kann das Verfolgen (englisch *tracking*) von Personen in der Umgebungskarte über die Zustandsübergänge von einer Sensoraufnahme zur nächsten modelliert werden.

Eine Implementierung auf dem BIRON Roboter der Universität Bielefeld [28] verwendet eine sogenannte Verankerung (englisch *anchoring*), um die symbolische Repräsentation von Personen und ihren Merkmalen mit den Sensorperzepten in Beziehung zu setzen. Einzelne Detektoren finden Beinpaare, Gesichter und Sprachquellen in der Umgebung des Roboters. Diese Einzelereignisse werden mittels eines Fusions-, Bewegungs- und Zusammensetzungsmodelles an ihren symbolischen Pendants verankert. Über die Zeit werden auf diese Weise die Personen und ihre symbolischen Zustände (stehen, laufen, sprechen etc.) ermittelt. Entsprechend kann der Roboter sich bei der Interaktion mit Menschen jeweils dem Sprecher zuwenden und diesen ansehen.

2.1.3 Verfahren

Aus der Vielzahl verschiedener Verfahren, die zum Finden von Personen mit mobilen Sensoren eingesetzt werden, haben wir schnelle und robuste Methoden ausgewählt und als getrennte Dienste im Sinne der serviceorientierten Architektur implementiert. Der PARTYBOT setzt den Laser und die Kamera als Sensoren ein, um Menschen in seiner Umgebung zu finden. Die Detektoren stehen als getrennte Dienste zur Verfügung, welche ihre Ergebnisse in Kugelkoordinaten relativ zur Kamera bereitstellen. Die Laserdaten und Navigation nutzen ebene Polarkoordinaten, sodass hier die Neigungskomponente schlicht entfällt. Die Einzeldetektionen werden zu Personen zusammengefasst.
Der Laser wird für Beinpaardetektion eingesetzt. Dabei werden zusammenhängende Segmente im Laserscan mit heuristischen Methoden als Beine klassifiziert. Danach wird versucht, eindeutige Paare zuzuordnen [29]. Auf die Kamerabilder wird der verbreitete Viola-Jones Algorithmus aus der *OpenCV* [39] Bibliothek zur Gesichtsdetektion angewendet. Außerdem verwenden wir die aktuell beliebten HOG Merkmale [13] zur Ganz- und Oberkörperdetektion im Kamerabild. Dazu haben wir eine eigene Implementierung erstellt und mehrfach mit selbsterstellten Testaufnahmen trainiert. Für den Echtzeitbetrieb auf dem PARTYBOT sind verschiedene Optimierungen nötig. So kann unter anderem die mögliche Position einer Person im Bild anhand der aus der Kameratelemetrie berechenbaren Position des Fußbodens eingeschränkt werden.
Die von den Sensoren erfassten Raumbereiche unterscheiden sich stark. Der Laser ist bis ca. 4m in 180° für die Beinpaardetektion nutzbar, das Sichtfeld der Kamera ist horizontal 48° und vertikal 36° (vgl. Abbildung 3). Die Kamera wird so bewegt, dass dieses optimal genutzt wird. Wenn der PARTYBOT auf der Suche nach Personen ist, sieht er sich mit Kameraschwenks um. Detektiert der Laser ein interessantes Objekt in der Nähe, wird die Kamera in Drehung und Neigung passend ausgerichtet, um nachzusehen, ob es sich dabei um eine Person handelt.
Die Funktion der einzelnen Detektoren wird in den folgenden Abschnitten erläutert. Im Anschluss wird die Funktion der Integration und Kamerasteuerung beschrieben.

2.2 Beinpaardetektion

Der von uns eingesetzte Beinpaardetektor verwendet die vom Roboter zur Verfügung gestellten Laserdaten. Ziel ist die Detektion von Personen im Halbfeld vor dem Roboter. Mit dem Beinpaarerkenner lässt sich so in einem Partyszenario sehr schnell die nähere Umgebung vor dem Roboter nach möglichen menschlichen Interaktionspartnern absuchen. Unser Ziel ist es, möglichst alle anwesenden Personen zu detektieren. Es sollen auch Detektionen von schlechter Güte berücksichtigt werden, um auf diese Weise möglichst viele Anfahrtziele zu erhalten.

2.2.1 Stand der Technik

Alle hier vorgestellten Verfahren verwenden Laserdaten, um Beine zu detektieren. In [62] wird ein Verfahren vorgestellt, das zylindrische Formen erkennen kann und damit auch Beine. Das Verfahren betrachtet dazu die Winkel zwischen den Linien, die ein Objekt umschließen. Wie unser Projekt basiert es auf dem Playerframework. Uns erscheint dieses Verfahren als ungeeignet, da Hosenbeine Falten werfen können und Beine somit beliebig weit von einer Halbkreisform entfernt sein können. Eine Detektion kann so in einigen Fällen nicht möglich sein. Wir verwenden hingegen die Varianz vom mittleren Abstand eines Objektes zum

Roboter. Eine einfache Beindetektion in diese Richtung wird in [5] aufgezeigt. Eine Erweiterung stellt das Verfahren dar, welches in [29] vorgestellt wird. Beide beschränken sich ausschließlich auf das Detektieren von Beinen. Es werden verschiedene Eigenschaften, wie z.B. die Beinbreite und der Abstand zwischen den Beinen, berücksichtigt, um diese von der Umwelt abzugrenzen. Die Auswertung verläuft mehrstufig. Zuerst werden die Segmente gebildet. Anschließend werden Segmente entfernt, die als Beine nicht in Frage kommen. Im letzten Schritt werden die möglichen Beine zu Personen gruppiert. Dieser Ansatz bildet die Grundlage unserer Implementierung. Ein weiteres mehrstufiges Verfahren wird in [3] vorgestellt. Hier werden ebenfalls nacheinander verschiedene Eigenschaften von Beinen berücksichtigt, um die Datenmenge einzuschränken. Parallel läuft hier ein Tracking, wobei Eigenschaften wie Schrittweite und Geschwindigkeit berücksichtigt werden.

2.2.2 Verfahren

Das verwendete Verfahren zur Beinpaardetektion mittels Laser orientiert sich am Verfahren, das in [29, S.64 ff] vorgestellt wird. Wir haben uns hier für dieses mehrstufige Verfahren entschieden, bei dem alle Klassifikationsschritte nacheinander abgearbeitet werden. Dies hat zum Vorteil, dass sich die Datenmenge stetig verkleinert und die komplexen Rechenoperationen nur noch auf wenige Elemente angewendet werden müssen. Ein Tracking wird an dieser Stelle nicht implementiert, da die detektierten Beinpaare nur als Eingabe für das Clustering (siehe 2.5) verwendet werden. Als Grundlage unserer Berechnungen dienen die Laserdaten des Halbfeldes vor dem Roboter mit einer Auflösung von 0,5° in einer Höhe von 40cm. Die Laserstrahlen werden also in der Regel unterhalb des Knies reflektiert und die Beine erscheinen näherungsweise halbkreisförmig. Es sollen jedoch auch Beine erkannt werden, wenn die Hose Falten wirft und von der Halbkreisform abweicht.

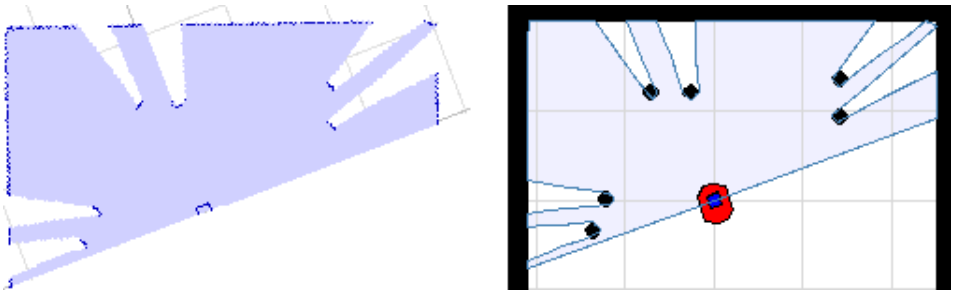


Abbildung 4: Extraktion der Segmente aus den Laserdaten

Nachdem die Laserwerte eingelesen wurden, werden sie segmentiert. Abbildung 4 zeigt die Segmentierung in einem gewöhnlichen Szenario mit drei Beinpaaren. Im linken Teil der Abbildung sieht man die entstehenden Segmente als dunkelblaue, zusammenhängende Teile. Die Elemente werden zu einem Segment zusammengefügt, wenn der Abstandswert zweier benachbarter Werte 7,5cm nicht überschreitet. Auf diese Weise bilden jene Elemente ein Segment, die vermutlich auch dasselbe Objekt repräsentieren. Der Schwellwert 7,5cm, sowie die folgenden Konstanten orientieren sich an den in [29, S.64ff] verwendeten Konstanten und wurden teilweise anhand der Ergebnisse der Evaluierung (siehe 5.5) angepasst.
Im nächsten Schritt werden die gefundenen Segmente klassifiziert, d.h. es werden die Segmente entfernt, die wahrscheinlich keine Beine sind. Zuerst werden dabei alle Segmente entfernt, die weniger als fünf

Elemente besitzen. Die nahegelegenen Segmente können keine Beine sein, weil sie zu schmal sind. Über die weiter entfernten Elemente können nur schlecht Aussagen getroffen werden, da weniger als fünf Elemente hierfür nicht ausreichend sind. Unterbrochene und unregelmäßige Strukturen könnten ansonsten als Beine erkannt werden. Man kann weder die Breite noch die Varianz zuverlässig bestimmen. Des Weiteren wird angenommen, dass Personen immer einen bestimmten Mindestabstand zu einer hinter ihnen liegenden Wand einhalten. Es wird geprüft, ob mindestens an einem Ende des Segmentes ein Abstand von mehr als 20cm zum angrenzenden Segment besteht. Um auch Beine schräg stehender Personen zu erkennen, werden auch Segmente als mögliche Beine erkannt, wenn an beiden Segmentgrenzen der Abstand größer als 5cm ist. Im Folgenden werden nur noch solche Segmente weiter betrachtet, deren Breite mindestens 8cm und höchstens 25cm beträgt, da in unserer Anwendung nur Beine dieser Breite erkannt werden sollen. Die Breite w(s) eines Segmentes s wird mit folgender Gleichung berechnet:

$$w(s) = 2\tilde{m}(s)\, tan(\alpha(s)) \tag{1}$$

Wobei $\tilde{m}$ der mittlere Abstand (Gleichung 2) des Segmentes zum Roboter und $\alpha(s)$ der Winkel (Gleichung 3) von einem Ende des Segmentes zum anderen ist.

$$\tilde{m}(s) = \frac{\sum_{k=i}^{j} m_k}{j - i + 1} \tag{2}$$

$$\alpha(s) = 0,5^\circ\ (j - i) \tag{3}$$

Die Differenz (j - i) gibt dabei die Anzahl der Elemente eines Segmentes an. Beim letzten Schritt der Klassifizierung wird die Standardabweichung (Gleichung 4) innerhalb eines Segmentes betrachtet. Wir machen uns hier die Eigenschaft von Beinen zu Nutze, dass sie annähernd halbkreisförmig sind. Sie sind weder komplett flach, noch besitzen sie eine hohe Standardabweichung von der mittleren Entfernung. Auf diese Weise können gerade Flächen und schräge Wände ausgeschlossen werden. In [29, S.64 ff] gibt der Verfasser an, dass sich Werte von weniger als 4cm für die Standardabweichung eignen, um Beine zu identifizieren. Unsere Beobachtungen haben jedoch gezeigt, dass ein Intervall von 1 bis 4,2cm besser geeignet ist, da insbesondere breite Hosen Varianzen knapp über 4cm erzeugen können, wenn diese Falten werfen. Zudem hat sich gezeigt, dass Beine in der Regel eine Varianz von mehr als 1cm haben. So können Wände aussortiert werden, die schräg oder senkrecht zu den Laserstrahlen verlaufen.

$$d(s) = \sqrt{\frac{\sum_{k=i}^{j} (m_k - \tilde{m})^2}{j - i + 1}} \tag{4}$$

Zuletzt wird die Gruppierung der Segmente zu möglichen Personen vorgenommen. Dabei werden zunächst für jedes Segment die Koordinaten des Schwerpunktes ermittelt. Hier ist zu beachten, dass nur der vordere Teil eines Beines erkannt wird, so dass man nicht den tatsächlichen Schwerpunkt des Beines, sondern einen nach vorne verschobenen Punkt erhält. Zwei Segmente werden nun zu einer möglichen Person, wenn der Abstand ihrer Schwerpunkte weniger als 50 cm beträgt. Der Algorithmus liefert als Ausgabe den Winkel und die Entfernung zum Gesamtschwerpunkt sowie die berechnete Güte für die Detektion. Es kann leicht

dazu kommen, dass ein Bein mehreren Personen zugeordnet wird. Ganz speziell können so aus drei Beinen zwei Personendetektionen entstehen. Wird ein Bein mehrfach zugeordnet, wirkt sich dies negativ auf die Güte aus. Die Güte wird mit Hilfe von Gleichung 5 berechnet. So erreichen Personen, die frei stehen und zum Roboter gerichtet sind oder Personen, die sich vom Roboter entfernen, besonders hohe Gütewerte. Die Variablen x_{links} und x_{rechts} geben an, wie vielen Personen das jeweilige Bein zugeordnet wurde. Der Faktor a ergibt sich aus der Entfernung der Beine und gewichtet Beinpaare höher, deren Abstand nahe am Mittelwert von 25cm liegt.

$$G(p) = \frac{a}{2}\left(\frac{1}{x_{links}} + \frac{1}{x_{rechts}}\right) \tag{5}$$

Insbesondere werden auch Segmente, die keinem anderen Segment zugeordnet werden konnten, zu einer Person; allerdings mit niedrigem Gütewert. Dies ist dann der Fall, wenn eine Person seitlich zum Roboter steht und das hintere Bein verdeckt ist.

2.3 Gesichtsdetektion

Natürlich ist davon auszugehen, dass der PARTYBOT auf jeder Party viele Blicke auf sich zieht. Damit er darauf passend reagieren kann, haben wir ihn mit einer Gesichtserkennung ausgestattet. Ein zugewandtes Gesicht zeigt aber auch allgemein Interesse und Bereitschaft zur Kontaktaufnahme. Gesichter sind auch seltener durch Kleidung oder andere Gegenstände verdeckt und häufig vollständig im Sichtfeld der Kamera. Der kleine Öffnungswinkel der Kamera und die Auflösung des Kamerabildes schränken die Detektion von Gesichtern ein. Dies wird zum Teil durch die Bewegung der Kamera und des Roboters kompensiert.

2.3.1 Stand der Technik

Die Suche nach Gesichtern in Bildern ist schon lange ein Gegenstand ausgiebiger Forschung [65]. Es existieren inzwischen einige erprobte und effiziente Verfahren welche u.a. in aktuellen Digitalkameras integriert werden [52]. Ein grundsätzlich wichtiger Aspekt ist die Möglichkeit der Detektion von nicht aufrechten und oder nicht zur Bildebene rotierten Gesichtern. Gesichtsdetektoren kann man, analog zur neuronalen Verarbeitung, in zwei Kategorien einteilen: Wissensbasierte und merkmalsgetriebene Verfahren.

Wissensbasierte Verfahren

Die *wissensbasierten* (englisch *top down*) Verfahren nutzen *a priori* bekannte heuristische Informationen über das gesuchte Objekt. Physiologisch werden diese Verfahren auch mit Blickbewegungen motiviert, welche ihrerseits die „interessanteren“(englisch *most salient*) Bildbereiche erfassen.
Typisch für solche Verfahren sind beispielsweise vektorisierte Schablonen oder Skelette, mit denen verglichen wird. Für Gesichtdetektion eignen sich diese nur begrenzt. Es muss eine „biegsame“ Schablone simuliert werden, die eine gewisse Flexibilität aufweist. Dem Problem wird mit Energieminimierungsalgorithmen und sogenannten Snakes[1] begegnet. Diesen Verfahren ist jedoch gemein, dass sie eine gesichtsnahe

[1] Snakes oder Aktive Konturen dienen dem Auffinden einer Kurve, die ein Energiefunktional minimiert, das ein Glattheits- und ein Approximationskriterium umfasst. Das Approximationskriterium ist dabei mit dem Bildinhalt verknüpft.

Initialisierung erfordern [65, S. 41ff.].

Des Weiteren werden auch Informationen wie Hautfarbe oder Form und Größerelationen des gesuchten Objektes verwendet. Die Nutzung von Farbwerthistogrammen zur Bestimmung von Bildbereichen mit Hautfarbe ist stark beleuchtungsabhängig und nicht sehr robust. Alternativ oder zusätzlich zur hautfarbenbasierten Detektion kann der Hintergrund mit Histogrammen oder aufgrund von Vorabinformationen geschätzt und vom Bild subtrahiert werden [16, S. 37ff.]. Nach der Anwendung von morphologischen Bildoperationen wie Closing ergeben sich zusammenhängende Bildbereiche welche wahrscheinlich zu Personen gehören. Diese Verfahren eignen sich gut für bekannte Räume und Beleuchtungsverhältnisse und werden meist im Zusammenhang mit Gestenerkennung verwendet [21]. Für die Gesichtserkennung kann man so die Menge der möglichen Positionen einschränken. Eine außergewöhnliche Anwendung stellt die künstlerische Installation „15 seconds of fame“ [55] dar, welche erkannte Gesichter für 15s in einem digitalen Bilderrahmen als Pop-Art-Gemälde anzeigt. Nach einer Beleuchtungskompensation werden Bildbereiche mit einem großen Anteil Hautfarbe selektiert. Danach werden Fehldetektionen mit einfachen Heuristiken unterdrückt. Dies gelingt nicht vollständig, so dass gelegentlich auch mal ein Arm oder eine Hand für 15s „berühmt“ werden.

Das Verfahren von Yang und Huang [64] ist auch im wörtlichen Sinn eine top-down Methode: Hier werden in verschiedenen Auflösungen diverse Regeln getestet. Sobald eine der Regeln nicht greift, wird das Bild verworfen. Das Bild wird durch Downsampling in eine Auflösungshierarchie überführt. Darauf werden drei Klassen von Regeln angewandt. Auf der gröbsten Stufe werden Regeln wie „die Mitte des Gesichtes hat weitgehend uniforme Helligkeit“ angewendet, um mögliche Gesichtspositionen zu finden. Nach einer grauwerthistogrammbasierten Normalisierung werden in feineren Auflösungen dann Kanten detektiert und mit Regeln verglichen. Auf der feinsten Stufe wird schließlich explizit das Vorhandensein von Gesichtsteilen wie Mund und Augen abgeprüft.

Merkmalsgetriebene Verfahren

Die zweite große Gruppe sind die merkmalsgetriebenen Verfahren (englisch *bottom up*), welche dichte Merkmale auf dem Bild berechnen. Physiologisch motiviert wird dieses Vorgehen durch die umfassende Bildverarbeitung im visuellen Kortex von Säugetieren [66], die auch ihrerseits eine Auswahl interessanter Bildregionen zur Klassifikation vornimmt (sogenannte bottom up Aufmerksamkeitsmechanismen).

Verschiedene Gesichtsdetektoren wurden mit künstlichen neuronalen Netzen realisiert. Interessant ist der Ansatz von Rowley et al. [49]. Hier wird in zwei Stufen vorgegangen. In der ersten bestimmt ein sogenanntes „router“ Netzwerk die Rotation des betrachteten Bildausschnittes X, in der zweiten wird das zur aufrechten Position rotierte Gesicht oder Nichtgesicht auf einen reduzierten Merkmalsvektor $G(X)$ abgebildet. Das „router“ Netzwerk berechnet aus einem 20x20 Pixel großen Bildausschnitt einen Winkel. Dazu werden 36 Ausgabeneuronen darauf trainiert, die Werte $\cos(\theta - 10^\circ \cdot n)$ für den Winkel θ anzunehmen. Für die Entscheidung wird jeder Wert eines Ausgabeneurons als Länge eines Vektors in Richtung $10^\circ \cdot n$ interpretiert. Der Winkel des Summenvektors gibt die Richtung an, in die das Eingabebild rotiert ist. Das Bild wird entsprechend rotiert, erneut normalisiert und dann von einem zweiten Netzwerk klassifiziert. Die Eingabe bilden 10x10 und 5x5 Pixel große Teilbilder sowie 20x5 Pixel Streifen [50]. Zum Training wird ein einfaches Bootstrapping-Verfahren verwendet. Es werden wiederholt Bilder ohne Gesichter als Eingabe verwendet und alle Detektionen als neue Negativbeispiele verwendet, um somit die Zahl der Fehldetektionen

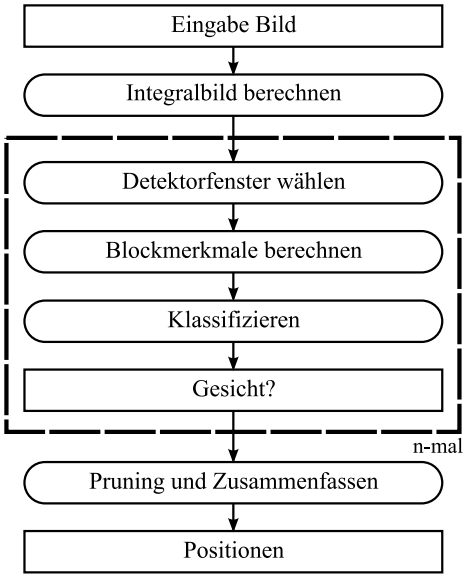
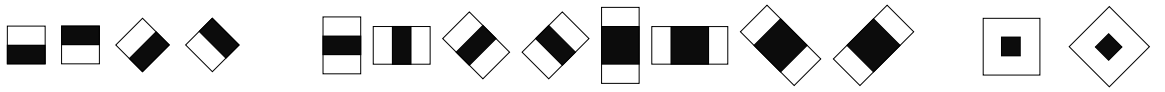


Abbildung 5: Arbeitsweise des Gesichtsdetektors

zu minimieren.

Ein anderer, etwas neuerer Ansatz ist die sogenannte „Synergistische Gesichtsdetektion“ [41]. Hier wird ein Faltungsnetzwerk verwendet, um Bilder in einen „Gesichtsraum“ abzubilden und danach eine Energiefunktion zu minimieren, welche die Korrespondenz von Gesichtsmerkmalen, Pose und Label abbildet. Das Faltungsnetzwerk nutzt eine spezielle, auf Bildverarbeitung ausgerichtete Architektur. Kleine, überlappende Bereiche aus dem Eingabebild werden jeweils zusammengefasst und als Eingabe für ein Neuron einer sogenannten Merkmalskarte (englisch *feature map*) verwendet. Die Berechnung ist vergleichbar mit der Anwendung eines kleinen Kernels und einer Sigmoidfunktion. Alle Neuronen einer Merkmalskarte haben dieselben Gewichte, so dass quasi eine Faltung der vorhergehenden Schicht mit einem Kernel stattfindet. Die 8 Merkmalskarten werden unterabgetastet und dann als Eingabe einer darauffolgenden Schicht von 20 Merkmalskarten verwendet. Hieraus werden 120 Faltungsausgaben erzeugt, welche schließlich von einer vollvernetzen Neuronenschicht auf 9 Werte abgebildet werden. Die weitere Auswertung kombiniert das Eingangsbild X mit einer Pose Z und dem Wert Y zu einer skalaren Energiefunktion $E_W(X, Y, Z)$. Der Wert Y (label) gibt an, ob es sich um ein Gesicht handelt. Dabei bildet die Pose, angegeben in Neigungs- und Drehungswinkel, dieses auf eine halbkugelförmige Gesichtmannigfaltigkeit $F(Z)$ ab. Die Bestimmung der gesuchten Werte Y^*, Z^* erfolgt über das Energieminimum $(Y^*, Z^*) = \mathrm{argmin}_{Y,Z} E_W(X, Y, Z)$. Dabei wird die Energiefunktion mit Gesichtsmerkmalsvektor $G(X)$ so definiert, dass ein effizientes Trainieren und Auswerten möglich ist. Mit einem konstanten Schwellwert T für Nicht-Gesichter wird hier $E_W(X, Y, Z) := Y||G(X) - F(Z)|| + (1 - Y)T$ definiert.

Ein wegen seiner hohen Geschwindigkeit und freien Verfügbarkeit oft eingesetztes Verfahren ist das von Viola & Jones [61]. Dies basiert auf einfachen Blockfeatures und einer mit dem Ada-Boost Algorithmus [14, S. 478] trainierten Detektorkaskade. Auch wir nutzen – aus den eben genannten Gründen – eine Weiterentwicklung dieses Verfahrens zur Detektion aufrechter Gesichter mit dem PARTYBOT. Das Verfahren wird im folgenden Abschnitt erläutert.



(a) Kantendetektoren (b) Liniendetektoren (c) Center-Surround

Abbildung 6: Vom Gesichtsdetektor verwendete Merkmale

2.3.2 Verfahren

Für die Gesichtsdetektion werden Bilder vom Kameraserver angefordert. Es werden bewusst nur Halbbilder verwendet, da anderenfalls Artefakte aufgrund des Interlacings auftreten. Auf diese Bilder wird das Detektionsverfahren von Viola & Jones in der Implementierung aus der *OpenCV*-Bibliothek [39] angewendet. Dies basiert auf der Weiterentwicklung von Lienhart et al. [32]. Die Detektion geht in fünf Schritten vor (Abbildung 5): Zunächst wird das sogenannte Integralbild berechnet. Über dies wird das Detektorfenster in verschiedenen Skalierungen geschoben. Auf diesem werden die Block-Merkmale bestimmt. Die Merkmale werden von einer Detektorkaskade klassifiziert. Die Detektionen werden im letzten Schritt, dem Clustering, zu Gruppen zusammengefasst und verschmolzen. Wir verwenden eine frei verfügbare Detektorkaskade für aufrechte, frontale Gesichter. Die minimale Gesichtsgröße und der Clustering-Schwellwert „minimale Unterstützung“ (englisch *min. support*) werden für den Einsatz auf dem SCITOS angepasst. Aus der Position und Größe der Detektion im Kamerabild wird die Position der Person im Raum geschätzt.

Merkmale

Zur effizienten Berechnung der Merkmale wird das Integralbild vorberechnet, welches an jeder Stelle die Summe aller Intensitätswerte der Pixel links und oberhalb von dieser enthält. Zur Berechnung 45°-rotierter Merkmale wird ein zweites Integralbild mit diagonalen Summierbereichen berechnet. Die eigentlichen Merkmale sind Differenzen von Summen über rechteckige Pixelbereiche (Abbildung 6). Diese lassen sich sehr effizient durch Subtraktion der Werte ihrer Eckpunkte im Integralbild berechnen. Es werden Kanten, Linien und Center-Surround bestimmt, indem jeweils eine Pixelsumme (weiß) von der anderen (schwarz) subtrahiert wird. Letztere bilden eine einfache Differenz von einem kleinen Bildbereich zu seiner Umgebung, wie dies auch bei dem gleichnamigen Verarbeitungsmechanismus auf der Retina geschieht [66]. Diese Merkmale sind offensichtlich gut für die Detektion von Augen oder Pupillen geeignet. Auch die Ecken- und Kantendetektoren sind an die Bildverarbeitungsmechanismen beim Menschen angelehnt [32, S. 1].

Die hier verwendete Menge stellt eine Erweiterung der ursprünglich von Viola und Jones verwendeten Merkmalsmenge dar, welche lediglich orthogonale Ecken und orthogonale, einfach breite Liniendetektoren verwendeten. Die 45°-rotierten, doppelt breiten Linien- und Center-Surround-Merkmale sind eine spätere Erweiterung von Lienhart et al., welche den Trainingsaufwand aber auch die Präzision des Detektors erhöht. Die ursprünglich von Viola und Jones verwendeten Merkmale entsprechen den Detailräumen erster und zweiter Stufe bei Multiskalenanalyse mit Haar-Wavelets [58], weswegen die Detektorkaskade bis heute oft auch als Haarkaskade (englisch *haar cascade*) bezeichnet wird.

Detektorkaskade

Um aus der Menge der Merkmale pro Detektorfenster eine Entscheidungsfunktion zu berechnen und gleichzeitig einen Klassifikator zu trainieren, wird eine Kaskade aus Klassifikatorensembles mit dem Ada-Boost Algorithmus [14, S. 478ff.] trainiert. Dieser bildet ein Ensemble aus sukzessive trainierten „schwachen" Klassifikatoren (Klassifikatoren mit knapp unter 50% Fehlerrate), wobei jeder spätere Klassifikator genau die Fälle gut entscheiden kann, bei denen die vorhergehenden falsch entschieden haben. Dazu werden die Trainingsbeispiele mit Gewichten versehen, die entsprechend der Leistung der Klassifikatoren angepasst werden. So werden schwierige Beispiele mit hohen Gewichten weitergereicht. Gleichzeitig werden aus dem Trainingsfehler Gewichte α_k für die schwachen Klassifikatoren berechnet. Nach dem Training entsteht so ein „starker" Klassifikator aus der gewichteten Summe der Einzelentscheidungen $h_k(x)$.

$$g(x) \quad = \quad \sum_k \alpha_k h_k(x) \tag{6}$$

Die so trainierten starken Klassifikatoren werden in einer Kaskade verschaltet, die sukzessive Detektorfenster zurückweist. Weitergereicht an die nächste Stufe werden jeweils die übrigen Bildausschnitte D_j. Die in der letzten Stufe nicht zurückgewiesenen Bildbereiche bilden die Menge D der Gesichtsdetektionen.

$$D_0 \quad := \quad \omega, \qquad D_j \quad = \quad \{x \in D_{j-1} \mid g_j(x) = 1\} \ \forall j \in 0..N, \qquad D \quad := \quad D_N \tag{7}$$

Von den frei verfügbaren Kaskaden wurde nach Vergleichstests eine gut funktionierende Kaskade ausgesucht (Abschnitt 5.3.2). In Anbetracht der guten Performanz sahen wir keine Notwendigkeit zum Trainieren einer eigenen Kaskade.

Clustering

Systembedingt treten überlappende Detektionen auf. Einzelne, nicht überlappende Detektionen sind oft Fehldetektionen (englisch *false positives*, kurz FP). Alle gefundenen Gesichter werden in Mengen von überlappenden Detektionen zusammengefasst. Befinden sich in einer Gruppe genug Detektionen, werden die Rechtecke zu einem Rechteck verschmolzen. Dieses stellt die endgültige Detektion eines Gesichtes dar. Kleinere Gruppen werden ignoriert.

Positionsschätzung

Anhand der Kameraposition und der Position im Bild kann der Winkel, in dem sich ein Gesicht befindet, bestimmt werden. Eine Entfernungsschätzung auf der Basis einer durchschnittlichen Gesichtshöhe von $\approx$ 0,21m wird nach Gleichung 8 berechnet.

$$d[\text{m}] \quad = \quad \frac{\text{Bildhöhe [Pixel]} \cdot \text{Gesichtshöhe [m]}}{\text{Fensterhöhe [Pixel]} \cdot \tan(\gamma/2)} \tag{8}$$

Evaluierungen mit Testaufnahmen belegen, dass für die minimale Gesichtsgröße unter 20 Pixel nicht weniger Gesichter gefunden werden (S. 82ff.). Damit ist die effektive maximale Entfernung $d_{\max}$, in der Gesichter gefunden werden, nach Gleichung 8 etwa 9m.

2.4 Ganzkörper- und Oberkörperdetektion

Damit der PARTYBOT nicht nur Personen in unmittelbarer Nähe bemerkt, sondern auch gezielt auf Menschen zufahren kann, haben wir zur Ergänzung der Detektoren für Gesichter und Beinpaare weitere für Ganzkörper und Oberkörper realisiert. Der Ganzkörperdetektor kann aufgrund der typischen Form eine entfernt stehende Person in einem hinreichend großen Abstandsbereich (4-18m) finden. Da der Fußbereich nicht immer im Bild ist und in Anbetracht des Einsatzszenarios auch mit hockenden und sitzenden Personen zu rechnen ist, wurde ein zweiter Parametersatz für Oberkörper bestimmt. Der Oberkörper- und der Ganzkörperdetektor werden parallel als OSGi-Dienste angeboten.

2.4.1 Stand der Technik

Bei der bildbasierten Detektion von Personen kann man anhand der Eingabedaten und verwendeten Merkmale drei Gruppen ausmachen: Wissensbasierte Verfahren, welche auf globalen Merkmalen in Einzelbildern basieren, Verfahren auf Differenzbildern oder optischem Fluss und schließlich merkmalsgetriebene Verfahren, welche informationsreiche Merkmale auf Einzelbildern berechnen.
Bei bekanntem oder leicht schätzbaren Hintergrund wird häufig mit Hintergrundsubtraktion (englisch *Background Subtraction*) und Segmentierung gearbeitet. Diese Verfahren werden oft mit globalen Merkmalen wie Hautfarbe und Konturen kombiniert. Auf einem mobilen Roboter lassen sich solche Verfahren nur adaptiv verwenden, indem die Bewegung der Kamera kompensiert und das Hintergundmodell angepasst wird [16, S. 37ff.].
Jabri et al. [24] verwenden beispielsweise zwei Hintergrundmodelle zur Subtraktion bei einer fixierten Kamera. Das erste Hintergrundmodell beschreibt die mittlere Farbintensität für alle drei Farbkanäle, das zweite wird aus den mit einem Sobel-Operator in allen drei Farbkanälen getrennt ermittelten Kanten bestimmt. Diese beiden Modelle werden getrennt vom Eingabebild subtrahiert, die kombinierte Differenzmaske wird aus dem Maximum über allen sechs Differenzen gebildet. Hier werden dann zusammenhängende Bereiche berechnet und schließlich deren Konturen extrahiert.
Diesem Vorgehen verwandt ist die zweite Gruppe von Verfahren, welche auf dem optischen Fluss oder Verschiebevektorfeldern operieren. Die zugrundeliegende Hypothese ist hier, dass sich der Hintergrund anders bewegt als die Personen, entweder durch Eigenbewegung der Personen oder Bewegungsparallaxe bei bewegtem Detektor. Im einfachsten Fall steht die Kamera still und bewegte Objekte sind größtenteils Personen. Hier ist das Hintergrund-/Vordergrundmodell durch ein Differenzbild direkt gegeben [21].
Die letzte, große Gruppe bilden die merkmalsgetriebenen Verfahren (englisch *bottom up*). Hier werden dichte, informationsreiche Merkmale auf dem Bild berechnet und dann mehr oder minder direkt mit einem Standardklassifikator als das gesuchte Objekt identifiziert. In letzter Zeit kommt hier oft die *Stützvektormethode* (englisch *support vector machine*, kurz SVM, siehe [9]) zum Einsatz.
Ein Beispiel ist das Verfahren von Papageorgiou et al. [44]. Hier werden Haar-Wavelets [58] auf dem Bild berechnet. Diese werden in horizontaler, vertikaler und diagonaler Richtung verwendet. Die Waveletberechnung wird von der vollständigen Repräsentation der Information bei der Multiskalenanalyse abgewandelt. So werden die feineren Detailstufen weggelassen, dafür aber die niederfrequenten in einem übervollständigen (englisch *overcomplete*) Raster berechnet. Mit großen Stichproben wird eine SVM trainiert, welche dann zur Detektion verwendet wird. Papageorgiou hat dieses Verfahren sowohl für Personen als auch für frontale

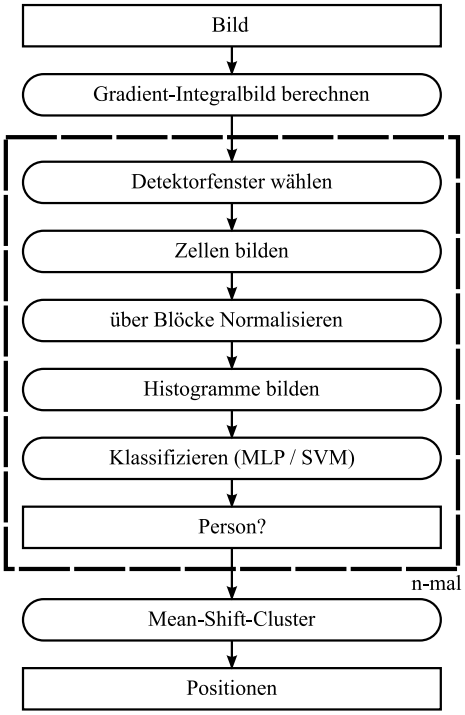


Abbildung 7: Arbeitsweise der HOG-basierten Detektoren

Gesichter verwendet [43]. Er bemerkt, dass sich in den Prototypen für Gesichter klar Details abzeichnen, während bei Ganzkörpern lediglich eine Silhouette erkennbar ist. Dies ist sicher damit zu erklären, dass die Intensitätsverteilung frontaler Gesichter recht einheitlich ist, während Personen in Kleidung und Haltung stark variieren.
Im Zuge des PASCAL[2] Programms sind verschiedene Untersuchungen zur Optimierung der Performanz lokaler Merkmale gemacht worden. Eine detaillierte Studie zur Optimierung von lokalen Merkmalen und SVM-Kernelfunktionen findet sich in [67]. Ein aktuell verbreitetes Merkmal sind die sogenannten orientierten Gradientenhistogramme (englisch *histogram of oriented gradients*, kurz HOG). Diese auf Gradienten basierenden Histogramme sind robust gegen kleine Konturänderungen, reagieren hingegen sensibel auf größere Beleuchtungs- und Farbänderungen und bleiben sehr unterscheidend für Objektkategorien [13]. Sie erzielen mit einer SVM gute Ergebnisse bei der Objektdetektion, wie Dalal et al. in der PASCAL Visual Object Classes Challenge 2006 demonstrieren konnten [15]. Basierend auf diesem Verfahren haben wir die Ganz- und Oberkörperdetektion auf dem PARTYBOT realisiert.

2.4.2 Verfahren

Für die Ganz- und Oberkörperdetektion werden Bilder vom Kameraserver angefordert. Es werden bewusst nur Halbbilder verwendet, da andernfalls Artefakte aufgrund des Interlacings auftreten. Als Merkmale werden gerichtete, normalisierte Gradientenhistogramme (HOGs) verwendet. Zur Klassifikation werden primär künstliche neuronale Netze eingesetzt, da diese ein gutes Laufzeitverhalten bei recht guten Ergebnissen

[2]EU gesponsertes „Network of Excellence“ on Pattern Analysis, Statistical Modelling and Computational Learning

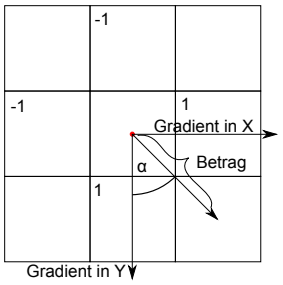
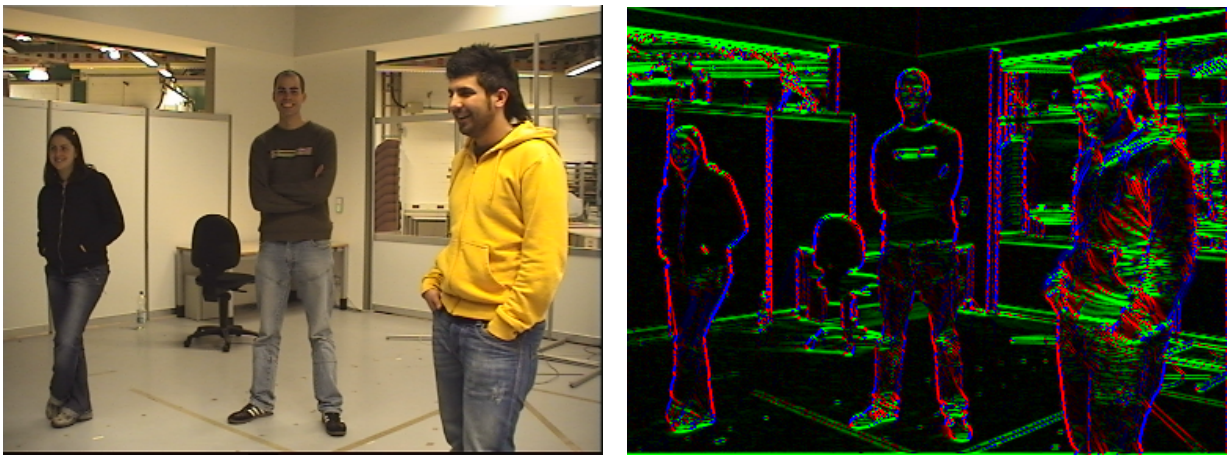


Abbildung 8: Berechnung des orientierten Gradienten in einer 3x3 Zelle.

zeigt. Zur Evaluierung und Bestimmung guter HOG-Parameter werden auch lineare Stütz-Vektor Maschinen (SVMs) verwendet, da diese bei kurzen Trainingszeiten repräsentative und reproduzierbare Ergebnisse bringen. Über ein Mean-Shift-Clustering werden die Detektionen bei erschöpfender Suche nachgefiltert. Im Laufe der PG stellte sich heraus, dass eine geschickte Einschränkung des Suchraumes möglich ist (Abschnitt 2.4.2, S. 29). Zur Wahl des Klassifikators sowie aller beteiligten Parameter werden mehrfach sukzessive Kreuzvalidierungen durchgeführt, diese sind im Detail im Abschnitt 5.2.3 beschrieben. Hier stellen wir die endgültig eingesetzte Methode im Einzelnen vor.

Orientierte Gradientenhistogramme

Im Folgenden wird ein kantenbasiertes Verfahren vorgestellt, das sich insbesondere zur Lokalisierung von Personen eignet [12]. Auf das Bild wird ein Gradientenoperator angewendet. Damit werden orientierte Gradienten berechnet, über die pro Bildbereich (Zelle) ein Histogramm über Winkelbereiche gebildet wird. Zusammenhängende Zellbereiche werden gemeinsam normiert. Das Vorgehen ist an die Bildwahrnehmung bei Säugetieren angelehnt, vor allem an die winkelsensitiven Zellen im V2 Areal des visuellen Kortex [66]. Abbildung 7 zeigt den allgemeinen Ablauf dieses Verfahrens. Zunächst werden die Bilder mit einem Kantenoperator gefiltert. Dies liefert ein Gradientenbild, das die Konturen der dargestellten Objekte hervorhebt. Ein Gradient ist ein Intensitätsunterschied zwischen benachbarten Pixeln. Der Gradientenoperator „durchläuft“ das Bild und berechnet anhand seiner Maske einen neuen Wert. Die Verarbeitung ist nicht auf Graustufenbilder beschränkt. Wird ein RGB-Bild als Eingabe erhalten, so wird der Gradient für jeden Farbkanal separat berechnet und der Farbunterschied mit der maximalen Differenz als Gradient benutzt. Es gibt verschiedene lokale Gradientenoperatoren [45].
Dalal et al. zeigten, dass ein einfacher, schnell zu berechnender eindimensionaler Filter $(-1, 0, -1)$ in diesem Anwendungsfall gute Ergebnisse liefert [12]. Der Filter wird in beide Richtungen des Koordinatensystem angewendet. Abbildung 9 zeigt ein so entstandenes Gradientenbild. Aus Effizienzgründen wird ein Integralbild der Gradienten gespeichert.
Bei der Berechnung der orientierten Gradienten aus dem Gradientenbild wird zudem der Betrag und Winkel des Gradienten bestimmt (vgl. Abbildung 8). Der Gradient kann vorzeichenlos (0° - 180°) oder vorzeichenbehaftet (0° - 360°) bestimmt werden. Wir verwenden vorzeichenlose Gradienten, da nicht a priori klar ist ob der Hintergrund oder die Person heller ist. Der Winkelbereich 0° bis 180° wird in 9 gleichgroße Abschnitte unterteilt ($\pm 10°$). Diese „Fächer “werden üblicherweise mit dem englischen Begriff „bins“ bezeichnet.



(a) Kamerabild (b) Gradientenbild

Abbildung 9: Gradientenbild einer Testaufnahme. Es wurden maximale Gradienten über alle Farbkanäle berechnet, in 9 bins zusammengefasst und jeweils 3 Richtungen in einem Farbkanal dargestellt.

Die Gradienten werden im nächsten Schritt zu Zellen zusammengefasst. In jeder Zelle werden die Beträge der Gradientenrichtungen nach Winkelbereichen aufsummiert. In jeder Zelle entsteht damit ein 1D-Histogramm der verwendeten Winkel-Einteilungen.

Um das Verfahren robuster gegenüber lokaler Beleuchtung und Schatten zu machen, werden die Zellen im Weiteren zu Blöcken zusammengefasst und anschließend wird innerhalb eines Blocks normalisiert. Wir verwenden rechteckige Blöcke. Die aus den Zellen entstehenden Blöcke werden überlappend berechnet, sodass jede Zelle zur Berechnung von mehreren Blöcken verwendet wird.

Als Normalisierungschema wird eine einfache L2-Norm verwendet. Diese wird auf jeden Block separat angewendet. Die L2-Norm berechnet sich ähnlich zum euklidischen Abstand aus:

$$x_i = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}} \tag{9}$$

Für einen Block $x = (x_1, \ldots, x_n)^T$ bestehend aus $n =$ (Zellhöhe x Zellbreite x #Bins) Werten wird jeder normierte Wert innerhalb des Blocks mit der obigen Formel berechnet. Wichtig ist, dass die Blöcke normalisiert werden. Die Unterschiede zwischen verschiedenen Normalisierungsschemata sind aber für Personen und natürliche Objekte recht gering [13, S. 6]. Die Blöcke stellen die Merkmalsvektoren für den anschließenden Klassifikator zur Verfügung. Der Merkmalsvektor ergibt sich also aus der Auflistung der Bins der einzelnen Blöcke.

Die HOGs werden grundsätzlich über Rechtecke mit einem festen Seitenverhältnis berechnet. Die Anzahl der Pixel der Zellen werden dabei entsprechend der Größe des Rechtecks angepasst. Dadurch werden die Merkmale invariant gegenüber der Skalierung.

(a) Gradientendurchschnitt (b) Gradientendurchschnitt (c) HOG

Abbildung 10: Durschnittsgradient in dreifarbiger und Schwarzweißdarstellung, zugehöriger Durchschnitts-HOG aus 7x15 Zellen ohne Überlappung und Block-Normierung

Klassifikation

Künstliche Neuronale Netze (kurz KNN, englisch *artificial neural networks*, ANN) versuchen den Lernprozess im Gehirn nachzuahmen. Bis heute wird üblicherweise das Perzeptron, welches Rosenblatt 1958 zur Nachbildung von Bildwahrnehmung einführte, verwendet [7]. Dabei werden gewichtete McCulloch-Pitts Neuronen mit einer linearen, stetigen und mehrfach differenzierbaren Schwellwertfunktion in einem gerichteten Graphen in mehreren „Ebenen“ verschaltet, daher spricht man oft auch von *multilayer perceptrons* (MLPs) . Zum Training wird eine Variante der sogenannte Backpropagation verwendet, bei der der Gradient des Fehlers am Ausgang rückwärts durch das Netz fortgepflanzt zur Modifikation der Gewichte verwendet wird, um per Gradientenabstieg ein (lokales) Optimum zu erreichen [6].
In der Evaluierung zeigte sich, dass die MLPs der FANN-Bibliothek wesentlich weniger Rechenzeit benötigen als die parallel eingesetzten SVMs (Abschnitt 5.2.3, S. 69ff.). Auch wenn rein lineare Separierung der HOGs bereits gute Klassifikationsergebnisse erzielt, wird die Qualität durch zwei Zwischenebenen (englisch *hidden layer*) noch gesteigert. Diese kommen mit sehr wenig Neuronen aus. Eine Steigerung der Neuronenzahl bringt keine Verbesserung. Im Endsystem werden MLPs mit 1.215 Eingängen gefolgt von Schichten aus 60 und 15 Neuronen mit zwei Ausgängen für Oberkörper und 1.701 Eingängen gefolgt von Schichten aus 84 und 21 Neuronen mit zwei Ausgängen für Ganzkörper eingesetzt.

Suchraum

Die Berechnung der HOGs wird in zwei Situationen benötigt. Zum einen beim Generieren der Trainingsdaten für den Klassifikator und zum anderen im Live-System bei der Klassifikationsentscheidung. Beim Generieren der Trainingsdaten werden annotierte Daten benötigt, in denen Personen und Nicht-Personen gekennzeichnet sind.
Im Live-System wird das Bild mit Hilfe einer Auflösungspyramide gescannt. Das Detektorfenster wird dabei

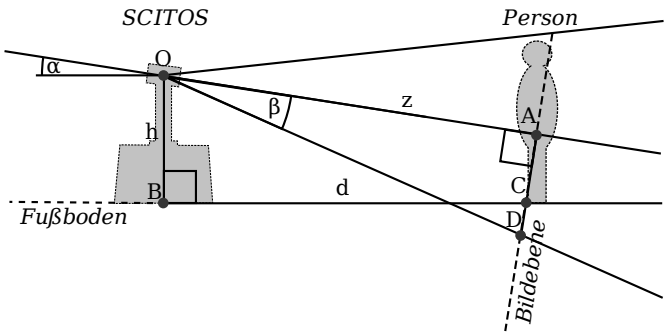


Abbildung 11: Abschätzung der Fußbodenhöhe

in dreidimensionaler erschöpfender Suche über das Kamerabild geschoben. Für verschiedene Entfernungen werden verschiedene logarithmisch äquidistante Skalierungen des Fensters verwendet und die HOGs entsprechend über unterschiedlich große Zellen berechnet. Aus der Evaluierung ergab sich, dass für 7-9 Tiefenebenen noch hinreichend viele Personen gefunden werden (S. 71ff.), mehr Ebenen hätten die Rechenzeit stark erhöht, weniger Ebenen führen zu deutlich weniger Detektionen. Basierend auf der Skalierung wird das Fenster um ganze Zellbreiten in horizontaler und vertikaler Richtung verschoben. Feinere Schrittweiten führen zu keiner wesentlichen Verbesserung, gröbere führen zum Übersehen von Personen.

Zur Einschränkung der im Bild zu untersuchenden Positionen nehmen wir an, dass die gesuchten Personen auf demselben Fußboden stehen wie der PARTYBOT. Aus der Position der Kamera im Raum lässt sich die Position des Fußbodens im Kamerabild berechnen. Entsprechend wird nach Personen nur in unmittelbarer Nähe des Fußbodens in nur wenigen vertikalen Positionen gesucht. Für Oberkörper wird ein Bereich für sitzende bis stehende Personen abgesucht, die oberen 2/3 des Personenfensters. So verbleiben nur 20-30% der möglichen vertikalen Positionen bei erschöpfender Suche. Die Fußbodenposition im Bild in einer Entfernung z lässt sich nach der Raumgeometrie wie folgt berechnen: Der Kameraserver liefert zu jedem Bild die Ausrichtung der Kamera (S. 117ff.), darunter den Neigungswinkel α. Die Kamera ist bei $O := (0, 0)$ in der Höhe $h = |\overline{OB}|$ (140cm) über dem Boden angebracht. Die Bildebene in Entfernung z schneidet den Fußboden im Punkt C. Bei festem Zoom hat das Blickfeld der Kamera den vertikalen Öffnungswinkel $\beta = 18°$ und die Unterkante des Bildes befindet sich bei D. Der Schnittpunkt C befindet sich auf den Geraden durch $\overline{AD}$ und $\overline{BC}$, also gilt Gleichung 10 und wir können $l = |\overline{AC}|$ direkt bestimmen (Abbildung 11).

$$\underbrace{\begin{pmatrix} z\cos\alpha \\ z\sin\alpha \end{pmatrix}}_{A} + l\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ h \end{pmatrix}}_{B} + d\begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{10}$$

$$\Rightarrow \qquad l[\mathrm{m}] = \frac{h - z\sin\alpha}{\cos\alpha} \tag{11}$$

Der Abstand z ergibt sich aus der Größe des Suchfensters und der geschätzten Größe der Person nach

α	0°	2°	4°	6°	8°	10°	12°
$d_0[m]$	4,00	3,57	3,22	2,92	2,67	2,44	2,25
$d_{\min}(1,5)[m]$	4,00	3,57	3,22	2,92	2,67	2,44	2,25
$d_{\min}(2,2)[m]$	4,00	3,57	3,22	3,75	4,64	6,04	8,52

Tabelle 1: Mögliche Personenentfernungen in Abhängigkeit vom Kamerawinkel

Gleichung 12, der Abstand der Fußposition $d = \overline{BC}$ ergibt sich daraus nach Gleichung 13.

$$z[\text{m}] = \frac{\cos\alpha \text{ Personenhöhe [m] Bildhöhe [Pixel]}}{\tan\beta \text{ Fensterhöhe [Pixel]}} \tag{12}$$

$$d[\text{m}] = \frac{z}{\cos\alpha} - h\tan\alpha \tag{13}$$

Des Weiteren ist natürlich die Höhe der Bildebene $n = |\overline{AD}|$ berechenbar.

$$n[\text{m}] = \frac{z\tan\beta}{2} \tag{14}$$

Somit ergibt sich $|\overline{CD}|$ in Pixeln nach Gleichung 15:

$$\text{Fußbodenhöhe [Pixel]} = \left(\frac{n-l}{n}\right)\frac{\text{Bildhöhe [Pixel]}}{2} \tag{15}$$

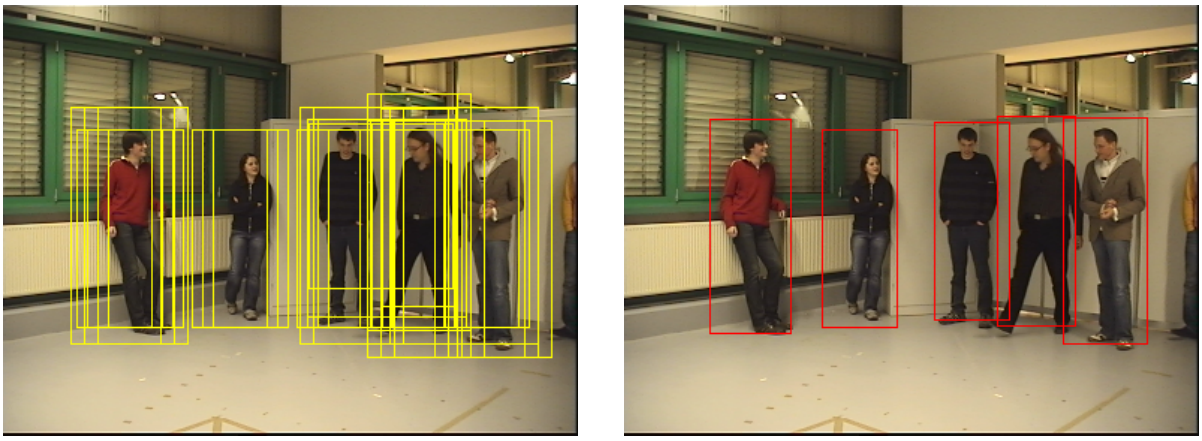
Tabelle 1 listet die möglichen Personenentfernungen in Abhängigkeit vom Winkel α. Die Entfernung, in der der Fußboden ins Bild kommt, wird mit d_0 bezeichnet. Die erste Entfernung mit vollständig sichtbaren Personen von 1,5m bis 2,2m über dem Fußboden ist als $d_{\min}$ aufgeführt. Das minimale Suchfenster von 92 Pixeln Höhe entspricht einer Personenhöhe von 2,2m (1,5m) einer Entfernung $d_{\max}$ von 20,77m (14,03m). Bei einer Neigung von über 13° ist ein so großer Teil der Bildebene vom Fußboden abgedeckt, dass darüber nicht mehr genug Raum für eine ganze Person ist (vgl. Abbildung 11).

Clustering

Da prinzipiell überlappende Detektionen auftreten, ist ein Clustering der Ergebnisse unvermeidlich. Wir verwenden den Mean-Shift-Cluster Algorithmus [11] zum Zusammenfassen benachbarter Detektionen, wie in Abbildung 12 dargestellt.
Mean-Shift ist ein unüberwachtes Clustering-Verfahren. Es ähnelt einem Gradientenabstiegsverfahren mit Verteilungsdichten. Es ist keine Annahme über die Verteilung notwendig. Mit Hilfe eines Kernel-Dichte-Schätzers wird die Wahrscheinlichkeitsdichte über den Merkmalsvektoren der Beobachtungen abgeschätzt. Dazu wird ein radialsymmetrischer Kernel wie z.B. der Gaußkernel K_G, oder der Epanechnikovkernel K_E für d Dimensionen verwendet. Letzerer beschreibt die Position in einer Hyperkugel mit Volumen V.

$$K_E(x) = \begin{cases} \frac{1}{2}V^{-1}(d+2)\,(1-||x||) & \text{für } ||x|| \leq 1 \\ 0 & \text{sonst} \end{cases} \tag{16}$$

$$K_G(x) = 2\pi^{-d/2}\exp\left(-\frac{1}{2}||x||^2\right) \tag{17}$$



(a) Überlappende Detektionen ohne Clustering

(b) Detektionsergebnis mit eingeschaltetem Clustering

Abbildung 12: Zusammenfassung benachbarter Detektionen

Die dazugehörigen Kernelprofile sind:

$$k_E(x) = \begin{cases} 1-x & \text{für } 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases} \tag{18}$$

$$k_g(x) = \exp\left(-\frac{1}{2}x\right) \tag{19}$$

Im Unterschied zu den Gradientenverfahren muss nicht zuerst die Verteilungsdichte und anschließend der Gradient über die Gradientenschätzung bestimmt werden. Der Mean-Shift-Vektor berechnet sich aus:

$$m_{\sigma,K}(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{\sigma}\right\|^2\right)} - x \tag{20}$$

Wobei n die Anzahl der Merkmalsvektoren x_i ist. Die Kernelbandbreite σ, sowie die Kernelfunktion K werden vorgegeben. Der Mean-Shift-Vektor berechnet sich damit aus der Differenz des gewichteten Mittelwerts, der anhand der Ableitung g des Kernelprofils k berechnet wurde, und der Mitte x des Kernels. Anschließend kann der Gradient mit Hilfe der Dichtefunktion bestimmt und der Kernel in die Richtung des Gradienten verschoben werden. Dabei ist die Bandbreite σ ein wesentlicher Parameter, dieser legt die Breite des Suchfensters fest. Je breiter dieser ist, desto größere Regionen werden zusammengefasst und desto schneller konvergiert der Algorithmus.

Der Gradient weist die selbe Richtung wie die Verschiebung des Mean-Shift-Vektors auf. Zudem muss die Schrittweite nicht angegeben werden, da sich diese aus dem Betrag des Mean-Shift-Vektors ergibt. Es ist also ein adaptives Verfahren, was ein weiterer Vorteil gegenüber dem einfachen Gradientenabstiegsverfahren ist, wo die Schrittweite festgelegt werden muss.

Das eigentliche Mean-Shift Verfahren startet nun mit einer vorgegebenen Anzahl von Startpunkten, die so verteilt sein sollten, dass der gesamte Merkmalsraum dadurch abgedeckt wird. Die Startpunkte müssen aber nicht gleichmäßig verteilt sein. Dann wird für jeden Startpunkt der Mean-Shift-Vektor berechnet und

der Startpunkt anhand dessen Richtung und Betrag verschoben. Dies wird für jeden Startpunkt solange wiederholt, bis der Gradient den Wert 0 erreicht, also das lokale Maximum findet. Es wird abgebrochen, sobald die Steigung unterhalb eines gewissen Schwellwertes liegt. Die daraus resultierenden Punkte sind die verschiedenen Klassenzentren des Clusterings. In einem letzten Schritt werden dicht beieinander liegende Klassen zusammengeführt. Dies wird meist in Abhängigkeit von der gewählten Kernelbandbreite gemacht. Als Merkmalsvektoren dienen bei uns die gefundenen Detektionen der Personen oder Oberkörper. Jede Detektion wird durch vier Werte, den Mittelpunkt des Rechtecks, repräsentiert durch eine X- und Y-Koordinate, und der Größe (Breite und Höhe) beschrieben. Diese Werte werden jeweils einem Merkmalsvektor zugeordnet. Das Mean-Shift-Clustering bestimmt nun die zusammengehörigen Detektionen und liefert nicht überlappende Rechtecke zurück. Die Kernelbandbreite ist dabei ein wichtiger Parameter, da dieser das Maß der Verschmelzung benachbarter Regionen bestimmt. Ist der Wert zu gering, so werden evtl. die zu einer Person zugehörigen Detektionen nicht zusammengefasst. Andererseits kann es bei zu großer Bandbreite passieren, dass die Detektionen zweier Personen zu einer zusammengeführt werden. Die Kernelbandbreite und weitere Parameter wurden während der Evaluierung (siehe Abschnitt 5.2.5) bestimmt.

Positionsschätzung

Anhand der Kameraposition und der Position im Bild kann der Winkel, in dem sich ein detektiertes Objekt befindet, bestimmt werden. Das Detektionsfenster umfasst die Ganz- und Oberkörper in den meisten Fällen recht zentral. Die Breite des sich aus dem Clustern ergebenden Rechtecks ist meist etwas schmaler als die Person. Eine sehr grobe Entfernungsschätzung auf der Basis einer durchschnittlichen Personenbreite von $\approx$ 0,5m wird nach Gleichung 21 mit dem Blickwinkel $\gamma = 48^\circ$ berechnet.

$$d[\text{m}] = \frac{\text{Bildbreite [Pixel]} \cdot \text{Personenbreite [m]}}{\text{Fensterbreite [Pixel]} \cdot \tan(\gamma/2)} \tag{21}$$

2.5 Integration

Die vier Detektoren werden als OSGi-Services mit einem gemeinsamen Interface angeboten. Alle geben einheitlich Detektionen, als Liste von Objekten desselben Typs, zurück. Dabei werden die Positionen in Kugelkoordinaten relativ zur Kamera angegeben. Tabelle 2 gibt eine Übersicht über die charakteristischen Eigenschaften der einzelnen Detektoren. Aufgrund dessen, dass die bildbasierten Detektoren einen kleineren Bereich als der Beinpaardetektor erfassen, wird die Kamera zum Aufstellen der Personenhypothesen über das Sichtfeld des Laserscanners geschwenkt. In Abschnitt 2.1.2 wurden Beispiele für symbolische und probabilistische Kombination isolierte Detektoren erwähnt [28, 35]. Wir verwenden empirisch ermittelte Genauigkeiten zum Zusammenfassen der Detektionen in Cluster. Dabei werden die Wahrscheinlichkeiten der beteiligten Detektionen kombiniert.

2.5.1 Kamerasteuerung

Da die Kamera des PARTYBOTS über eine Motorsteuerung verfügt, liegt es nahe, sich über eine sinnvolle Steuerung zur Ausrichtung Gedanken zu machen. Mit einer festen vertikalen Achse, mit einem Drehbereich von -180° bis 180°, und einer mitdrehenden horizontalen Achse, die von -30° bis 90° gedreht werden kann,

Detektor	$d_{\min}$ [m]	$d_{\max}$ [m]	Δd	FOV	Δ°	Zeit
Beinpaare	0,5	3,0- 5,0	1%	± 90	$\leq 2^\circ$	0,12 s
Gesichter	0,5	9,0	20%	$\gamma \pm 24$	$\leq 1^\circ$	0,35 s
Oberkörper	1,2- 3,0	14,0-20,0	250%	$\gamma \pm 22$	$\leq 5^\circ$	0,57 s
Ganzkörper	3,0- 4,0	14,0-20,0	200%	$\gamma \pm 22$	$\leq 5^\circ$	0,81 s

Tabelle 2: Arbeitsbereich, Schätzfehler und Rechenzeit der einzelnen Detektoren. γ ist der Drehungswinkel der Kamera. Die Bereiche der Ober- und Ganzkörperdetektor sind abhängig von Kameraneigungswinkel und Personengröße.

besitzt die Kamera eine hohe Beweglichkeit. Jedoch ist der Bereich, den die Kamera überprüfen kann, mit einem Öffnungswinkel von 48° zu dem 360° großen Bereich um den Roboter, in dem sich Personen aufhalten könnten, begrenzt. Es gibt mehrere Strategien um diese Situation zu bewältigen, wovon eine das schrittweise Abtasten des zu kontrollierenden Bereiches ist und eine weitere das Überprüfen von Detektionen anderer Detektoren. Beide Strategien sind in das CamScan-Bundle des PARTYBOTS eingeflossen. Dieses überwacht die Detektionen des Beinpaardetektors, welcher mit einem 180° großem Bereich die Hälfte des zu kontrollierenden Bereiches abdeckt, und richtet die Kamera um die vertikale Achse auf die mögliche Person aus. Sollten keine Ergebnisse des Beinpaardetektors vorhanden sein wird die Kamera in 40° Schritten abwechselnd im und gegen den Uhrzeigersinn gedreht. Bei der Ausrichtung der horizontalen Achse ist die Distanz zur Person entscheidend. Sollte sich diese in einem nahen Bereich bis zu 3m aufhalten sollte die Kamera höher schauen um das Gesicht zu erfassen. In diesem Bereich können auch keine ganzen Personen gefunden werden, da diese den aufgenommenen Bildausschnitt überragen würden. Je weiter die Person entfernt ist, desto mehr ist von dieser im Bildausschnitt zu sehen und die Kamera kann sich in einem niedrigeren Winkel ausrichten. Die Ausrichtung der horizontalen Achse wird nach Formel 22 berechnet.

$$\text{tilt}(d) = \begin{cases} 6(2-d) & \text{für } d < 3 \\ -6 & \text{sonst} \end{cases} \tag{22}$$

2.5.2 Zusammenfassen der Detektionen

Die vier Detektoren liefern unabhängige Ergebnisse. Durch das Zusammenfassen sollen Detektionen, welche die selbe Person wiedergeben, kombiniert werden. Korrekte Ergebnisse werden so im günstigsten Fall von anderen Detektoren bestätigt.

Für jeden Detektor wird ein Toleranzbereich und eine Gewichtung definiert. Der Toleranzbereich gibt die Genauigkeit der Positionsangaben der Detektion wieder. Die Gewichtung eines Detektors geht in die Berechnung des Konfidenzwertes eines Detektionsclusters ein. Zusätzlich wirkt sich die Anzahl der Detektionen innerhalb einer zusammengefassten Gruppe auf diese Berechnung aus. Weiterhin existieren die Möglichkeiten einen Detektor für die Entfernungsberechnung zu bevorzugen sowie die Berechnung der Detektionscluster auf einen Detektionstyp zu stützen. Diese Konfigurationswerte liefern die Basis für das Zusammenfassen von Detektionen.

Der Clusteringalgorithmus bekommt als Eingabe eine Liste aktueller Detektionsobjekte der Detektoren. Zwischen diesen Detektionen werden anhand der definierten Toleranzbereiche Beziehungen ermittelt. Es

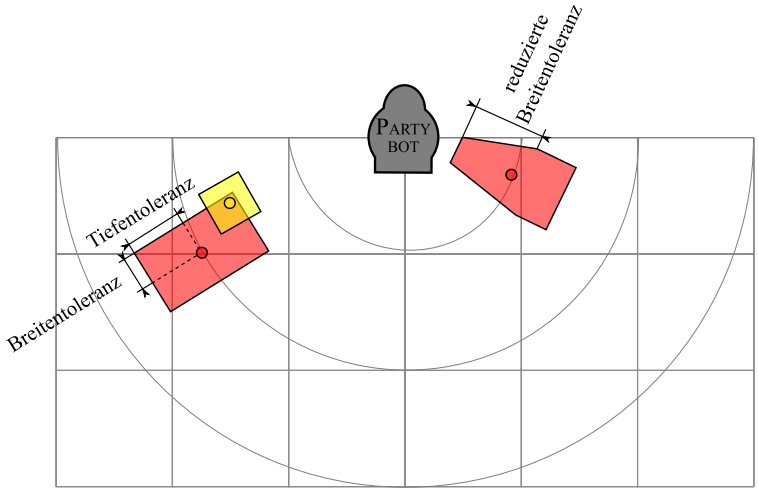


Abbildung 13: Toleranzen und zusammenfassbare Detektionen beim Clustering

besteht eine einseitige Beziehung zwischen zwei Detektionen, wenn die Koordinaten der ersten Detektion im Toleranzbereich der Zweiten liegt. Der Toleranzbereich wird über drei Werte definiert. Zum einen bezieht sich die Tiefentoleranz auf die Entfernungsangaben, weiter nimmt die Breitentoleranz Einfluss auf die Breite des Bereiches. Ein weiterer Schwellwert gibt eine Entfernung vom Roboter an, von der an die Breitentoleranz zum Roboter hin abnimmt. Auf diese Weise ist nahe am Roboter die Winkeldifferenz zwischen zwei Detektionen relevanter als die Breitentoleranz.

Für die Bildung von einzelnen Detetektionsgruppen, den Clustern, werden nun die Beziehungen zwischen den Detektionen untersucht. Hier werden verschiedene Fälle betrachtet. Falls eine Detektion nicht in Beziehung mit einer Anderen steht, sowie keine andere Detektion in deren Toleranzbereich liegt, wird aus ihr ein eigener Cluster gebildet, der keine weiteren Detektionen enthält. Existiert eine abgeschlossene Gruppe, so dass innerhalb dieser Gruppe jede Detektion mit jeder Anderen in Beziehung steht, dann wird diese Gruppe von Detektionen zu einem Cluster zusammengefasst. Weitere Cluster werden gebildet, indem zu einer Detektion weitere Detektionen gebunden werden, die mit dieser in Beziehung stehen. Hierbei kann es durch große Toleranzbereiche zu Überschneidungen kommen, so dass mehrere Detektionen im möglichen Bereich liegen. In diesem Fall kann die Clusterzuordnung nach Regeln entschieden werden. In die Entscheidung kann die Entfernung zwischen den Detektionen, die Konfidenz sowie die Art des Detektors mit einfließen. Sollten Überschneidungen existieren, wird zunächst eine Menge von Detektionsgruppen gewählt, welche als Basis für die Clusterbildung dienen. Diese Menge umfasst alle einelementigen Gruppen von Detektionen, mit denen keine andere in Beziehung steht, sowie mehrelementige Gruppen von Detektionen, die gegenseitig in Beziehung stehen. Diesen Gruppen werden dann jeweils die Detektionen hinzugefügt, in deren Toleranzbereich genau nur diese Gruppe liegt. Detektionsobjekte, welche mehreren Gruppen zugeordnet werden können, werden dem Cluster hinzugefügt, dessen Mittelpunkt am nächsten liegt.

Wird die Berechnung der Cluster auf einen Detektionstyp gestützt, so werden zunächst alle Detektionen dieses Typs ermittelt und nach absteigendem Konfidenzwert sortiert. Anschließend werden zu diesen bevorzugten Detektionen diejenigen Detektionen von anderem Typ ermittelt, die in Beziehung mit diesen stehen, und zu einem Cluster zusammengefasst. Auf diese Weise wird sichergestellt, dass zuerst die relevanteren

Detektionen mit höherer Konfidenz bearbeitet werden. Zusätzlich beinhaltet jedes Cluster genau eine Detektion des bevorzugten Typs.
Für jedes Cluster wird ein Mittelpunkt und eine zusammengefasste Konfidenz aus den beinhalteten Detektionen ermittelt. Der Mittelpunkt liefert dann die Position der detektierten Person. Er berechnet sich jeweils aus dem Durchschnitt des Entfernungswertes und des Drehwinkels. Sollte über die Konfiguration gefordert sein, dass bei der Entfernungsberechnung ein Detektor bevorzugt wird, so wird hier nicht der Durchschnitt berechnet. Beinhaltet ein Cluster eine Detektion dieses bevorzugten Detektortyps, so wird der Entfernungswert von dieser Detektion übernommen. So kann ein Detektor, der besonders gute Entfernungsmessungen liefert, favorisiert werden. Ein Clusterobjekt kapselt die zugehörigen Detektionen und liefert zusätzlich Methoden zur Berechnung eines zusammenfassenden Konfidenzwertes und der Position. Der kombinierte Konfidenzwert eines Clusters wird als gewichteter Mittelwert der einzelnen Detektionen berechnet. Die Gewichtung wird über den Detektortyp in der Konfiguration festgelegt. Zusätzlich geht die Clustergröße in die Berechnung mit ein. Cluster mit nur einem Element werden abgewertet und solche mit mehreren Detektionen entsprechend aufgewertet. Ein Clusterobjekt lässt sich wiederum in ein Detektionsobjekt umwandeln, so kann der Clusteringalgorithmus als Detektor ausgewiesen werden. Hierbei gehen aber Informationen über die einzelnen zusammengefassten Detektionen verloren. Die Anzahl der in einem Cluster vorhandenen Elemente spiegelt sich dann nur noch im berechneten Konfidenzwert wider.

3 Navigation in dynamischen Umgebungen

Mobile Roboter agieren häufig in hochgradig dynamischen Umgebungen, wie z.B. Wohnungen, Museen oder Veranstaltungen, bei denen höchstens eine statische Umgebung bekannt ist und müssen in dieser zuverlässig navigieren. Die Besonderheit liegt in der Dynamik der Hindernisse wie Personen, Türen oder auch Tischen, und der daraus resultierenden Problematik, dass auf diese Objekte bei der Planung oder während der Fahrt reagiert werden muss. Da die übliche Umgebung des PARTYBOT dynamisch ist, muss ein besonderes Augenmerk auf die Navigation unter Berücksichtigung dynamischer Hindernisse erfolgen. Als Beispiel kann man sich, wie in Abbildung 14 zu sehen, einen Raum mit zwei Ausgängen vorstellen.

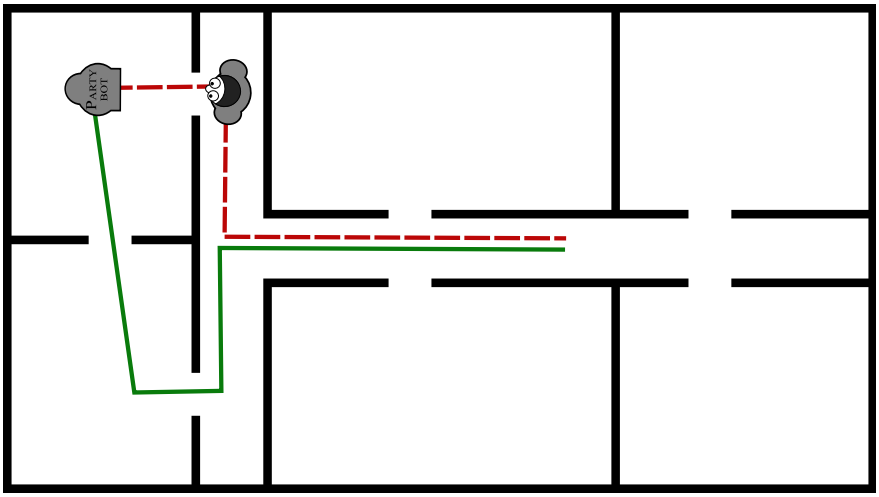
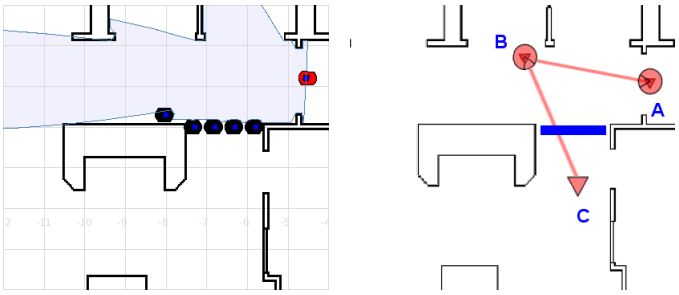


Abbildung 14: Problemstellung: dynamisch blockierter Pfad (rot gestrichelt), alternativer Pfad (grün)

Die eine Tür ist blockiert (roter Pfad) und der Navigationsalgorithmus plant den Weg zunächst durch diese Tür, da er nur eine statische Karte seiner Umgebung besitzt. Der Roboter wird aller Wahrscheinlichkeit nach vor der blockierten Tür stehen bleiben bis diese wieder geöffnet wird. Dies beruht auf der Tatsache, dass die Route mit Hilfe einer statischen Karte seiner Umgebung erstellt wird und diese keine dynamischen Informationen enthält. Unser Ziel war es, eine dynamische Karte mit Hilfe des Lasers und der aktuell lokalisierten Position zu erstellen. Auf dieser soll der Navigationsalgorithmus des Player-Frameworks (siehe D.3) angewendet werden, sodass der Roboter seinen Weg umplanen kann (grüner Pfad). Des Weiteren soll ein OSGi-Service implementiert werden, welcher die Entscheidung über eine Neuplanung trifft.

An dem folgenden Beispiel, welches am Ende von 3.2.1 nochmals aufgegriffen wird, sollen sowohl die Motivation für die Entwicklung als auch die Funktionsweise des Dynamicmapping-Verfahren veranschaulicht werden. Sofern mehrere Wege zu einem Ziel führen, wäre es wünschenswert, dass der Roboter einen anderen Weg plant, sobald er bemerkt, dass der zuerst gewählte Weg versperrt ist.

In Abbildung 15 liegt eine solche Situation vor. Der Roboter soll vom Punkt A zum Punkt B fahren und anschließend zum Punkt C. Auf der Karte gibt es einen direkten Weg zwischen B und C, dieser soll jetzt aber versperrt sein (blaues Rechteck). Der Roboter fährt nun nicht mehr den direkten Weg, sondern plant seinen Weg links außen vorbei.



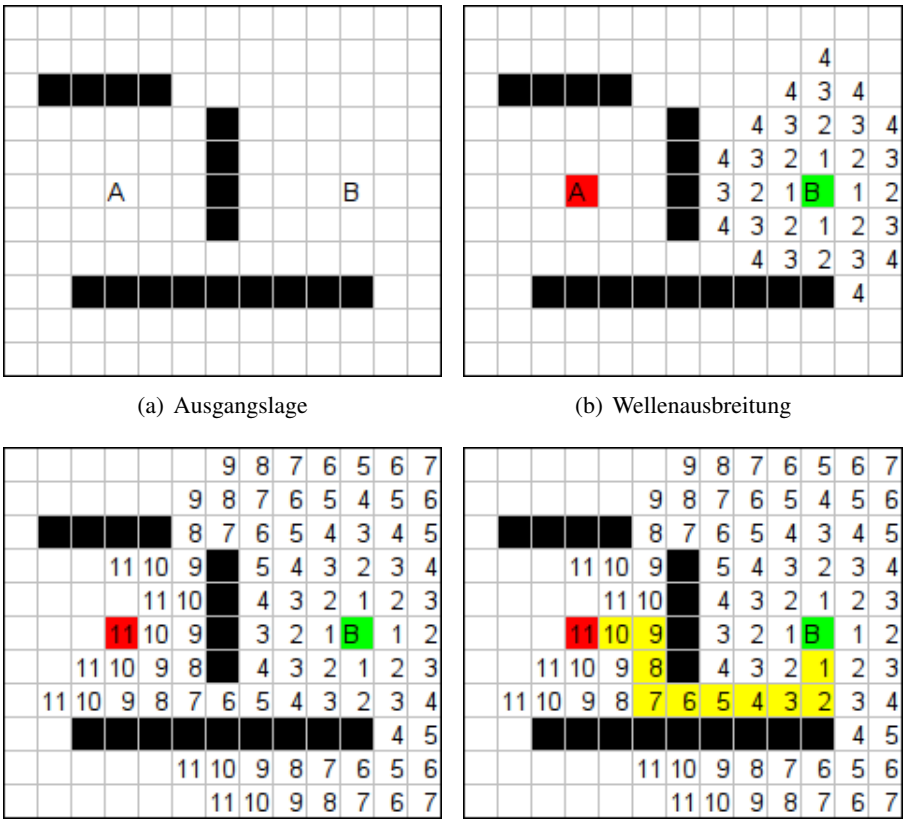
(a) Ausgangslage (b) Aufbau

Abbildung 15: Ausgangslage des Beispielszenarios

3.1 Stand der Technik

Der in [34] vorgestellte Algorithmus kann ohne Karte in dynamischen Umgebungen navigieren. Dazu wird eine Gerade zwischen Start- und Endpunkt gelegt und diese abgefahren bis der Roboter sein Ziel erreicht oder auf ein Hindernis, welches er mit seinen Sensoren detektiert hat, trifft. Im Falle eines Hindernisses fährt der Roboter an der Kontur des Hindernisses entlang bis er wieder die Gerade zwischen Start- und Zielpunkt erreicht. In [1] wird ein Verfahren beschrieben, welches eine auf Fuzzy-Logik basierende Hindernisvermeidung bietet. Es wird eine autonome Navigation in dynamischen Umgebungen bereitgestellt, wobei keine Informationen über die Umgebung benötigt werden. Detektierte Hindernisse werden in einer dynamischen Karte übertragen. Bereits erkannte Hindernisse altern und somit wird nur eine relativ lokale Umgebung des Roboters betrachtet. In [33] wird eine Methode demonstriert, bei der mit Hilfe von Potentialfeldern in komplett unbekannten Umgebungen navigiert werden kann. Das Ausweichen von dynamischen Hindernissen wird dabei durch ein Potentialfeld, welches dem menschlichen Immunsystem nachgebildet ist, erzielt.

Ein erprobter Algorithmus zur Pfadplanung auf statischen Karten wurde in [31] vorgestellt. Er basiert auf einer statischen Rasterkarte seiner Umgebung und wird benutzt, um in engen Labyrinthen einen Pfad zwischen zwei Punkten zu planen. Die grundlegende Funktionsweise lässt sich mit der Ausbreitung von Wellen vergleichen. Daher wird der Begriff Wavefront oft als Synonym für diesen Algorithmus verwendet. In Abbildung 16 ist der Algorithmus an einem einfachen Beispiel demonstriert. Der Algorithmus löst das Problem der Wegfindung zwischen zwei Punkten (Abbildung 16(a)), indem einer der beiden Punkte als Ziel- und der Andere als Startpunkt markiert wird. Nun werden vom Zielpunkt aus wellenförmig alle umliegenden Rasterzellen mit einem höheren Wert belegt (Abbildung 16(b)). Dieses Verfahren wird nun für bereits markierte Rasterzellen wiederholt. Diese wellenartige Ausbreitung bricht ab, sobald der Startpunkt erreicht wird (Abbildung 16(c)). Nun muss in einer Backtracking-Phase ein Weg bestimmt werden. Dazu wird der Weg immer in Richtung absteigender Werte geplant. Um möglichst effiziente Wege zu planen, werden solche mit möglichst wenig Knicken gesucht (Abbildung16(d)). In das für den PARTYBOT genutzte Player-Framework ist dieser einfache Algorithmus bereits integriert. In der verwendeten Version werden auch diagonale Bewegungen berücksichtigt. Zwar lässt sich dynamisch eine neue Karte einlesen, aber diese enthält keine dynamischen Informationen, die für einen Einsatz in einer dynamischen Umgebung unabdingbar sind.



(a) Ausgangslage (b) Wellenausbreitung

(c) Start erreicht (d) Backtracking

Abbildung 16: Funktionsweise des Wavefront-Algorithmus

3.2 Dynamische Wegplanung

Der Algorithmus zur dynamischen Wegplanung wurde als Treiber in das Playerframework (siehe D.3) integriert. Er benutzt eine vorhandene Karte (Mapfile), seine geschätzte Position (AMCL), den Laser (Sicks300) und die Odometriewerte (SCITOS-Treiber). Der Treiber stellt eine Karte zur Verfügung (Abbildung 17). Somit können beliebige Navigationsalgorithmen mit unserem Dynamicmapping-Treiber zusammen arbeiten.

Zur Erstellung der dynamischen Karte wird eine Karte der Umgebung benötigt, wie sie auch vom Wavefront-Algorithmus benutzt wird. Sie wird vom Treiber Mapfile zur Verfügung gestellt. Zusätzlich werden die Laserwerte des Halbfeldes vor dem Roboter mit einer Auflösung von 1° verwendet. Erkannte Hindernisse werden in einer internen Karte vermerkt. Zur Erzeugung der dynamischen Karte werden die Umgebungskarte und die interne Karte verschmolzen (Abbildung 18). Anschließend wird die resultierende Karte normalisiert, d.h. das Raster enthält daraufhin nur noch die Zahlen 1 und -1, Wobei 1 nicht passierbar und -1 frei bedeutet. Diese Karte wird nun dem Treiber Wavefront übergeben, der dann den Weg um die dynamisch eingetragenen Hindernisse herum planen kann.

Die Entscheidung, ob ein ursprünglich geplanter Weg verworfen und ein neuer geplant werden soll, wird in einem OSGi-Service (siehe 3.2.2) getroffen.

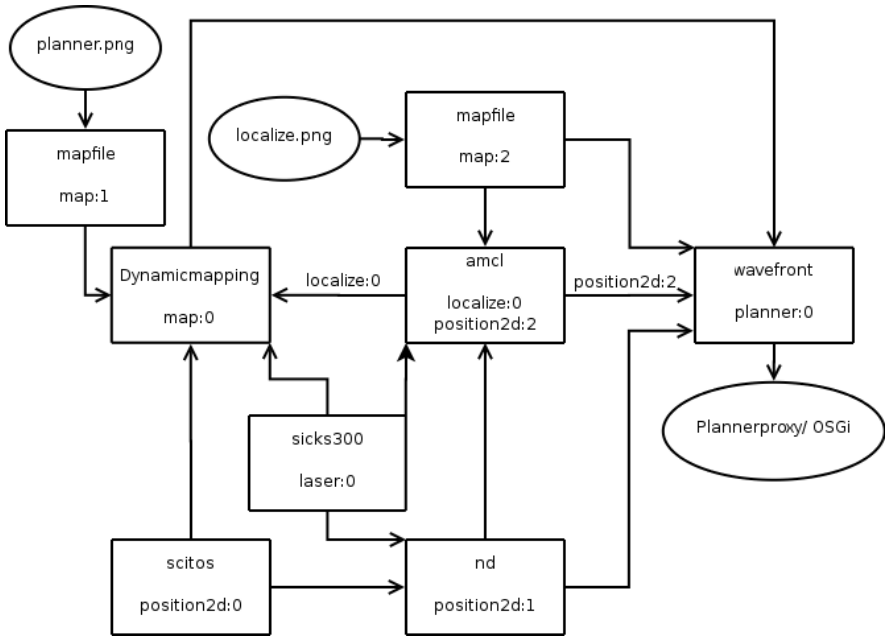


Abbildung 17: Integration des von uns entwickelten Treibers Dynamicmapping in das Player-Framework

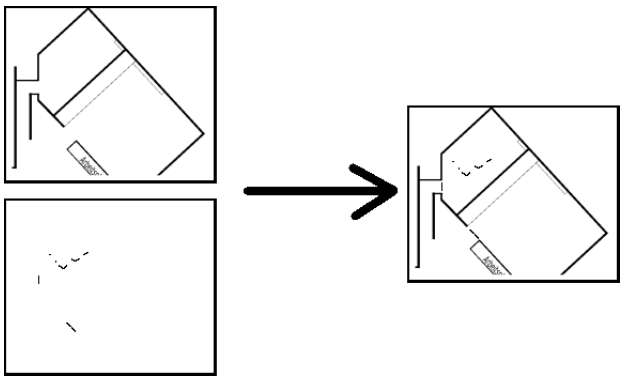


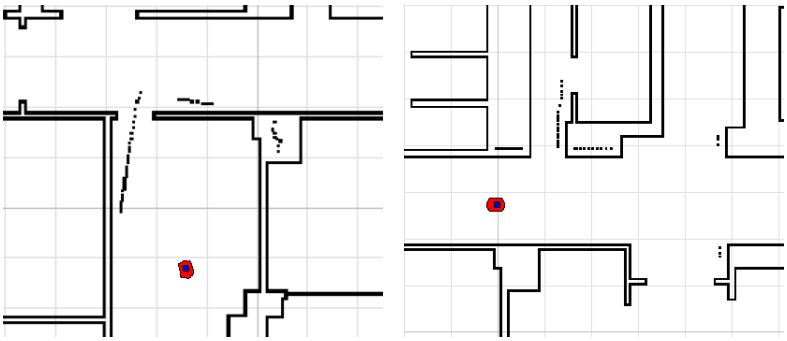
Abbildung 18: Verschmelzen der Umgebungskarte und der dynamischen Karte mit anschließender Normalisierung

3.2.1 Treiberimplementierung

Der Algorithmus, der die dynamische Karte erstellt, wurde in einem Thread implementiert. Die *Sleeptime* des Threads kann über einen Parameter (sleeping_time siehe C.1) angepasst werden, damit dieses rechenintensive Verfahren nicht die gesamte Rechenleistung beansprucht. Zunächst wird unterschieden, ob die Winkeländerung im Vergleich zum letzten Durchlauf unter einer gewissen Grenze (max_angle siehe C.1) liegt. Ist dies nicht der Fall, so werden die Werte im aktuellen Durchlauf nicht berücksichtigt. Da sie in der Regel wegen der Asynchronität zwischen Laserwerten und Lokalisation unbrauchbar sind, würden ansonsten Objekte falsch eingezeichnet werden. Mit Asynchronität ist an dieser Stelle der Effekt gemeint, dass die Lokalisation sich nicht so schnell aktualisiert wie der Laser und es so zu einem Zeitversatz kommt. Falls die Werte klein genug sind, wird der eigentliche Haupteil des Algorithmus durchlaufen. Zunächst altern die Werte aus den vorherigen Durchläufen. Die Laserwerte werden in einem Array mit 361 Werten zur Verfügung gestellt. Wir verwenden jedoch nur jeden Zweiten, da diese Auflösung für unsere Zwecke absolut ausreichend ist und so der Rechenaufwand um die Hälfte reduziert werden kann. Für die weitere Berechnung werden nur Entfernungswerte berücksichtigt, die innerhalb eines gewissen Bereiches liegen, der mit Hilfe der Übergabeparameter (max_dist und min_dist siehe C.1) gewählt werden kann. So kann sich der Roboter nicht selbst einzeichnen und der Bereich, in dem der Laser zuverlässig arbeitet, kann abgegrenzt werden. Die Karte entspricht in der internen Darstellung einem Byte-Array. Sie ist mit der linken unteren Ecke beginnend zeilenweise in das Array eingetragen. Die Stellen im Array können dabei Werte zwischen -127 und 127 annehmen.

Der Algorithmus durchläuft eine Schleife, in der die Laserwerte wie folgt ausgewertet werden. Zunächst wird getestet, ob der Abstandswert, den der Laser an der jeweiligen Position misst, innerhalb der über Parameter (max_dist und min_dist siehe C.1) gesetzten Grenzen liegt. Mithilfe von Rotation und Translation wird die Position des jeweiligen möglichen Hindernisses in der Karte bestimmt. Nun wird an dieser Stelle und im Umkreis überprüft (scan_depth siehe C.1), ob ein festes Hindernis in der Eingabekarte eingezeichnet ist. Ist dies der Fall, wird der entsprechende Pixel mit diesem Hindernis gleichgesetzt und nicht weiter beachtet. Andernfalls wird an dieser Stelle der Wert in der Karte erhöht. Wurde ein Hindernis mehrfach gesichtet und der Wert an der Position ist dementsprechend hoch, so wird die Zahl schneller hochgezählt. So bleiben langsame dynamische Hindernisse und unbekannte statische Hindernisse länger an der beobachteten Position eingezeichnet. Die Zahl, bis zu der höchstens hochgezählt wird, kann ebenfalls gewählt werden (upper_bound siehe C.1). Sie kann jedoch höchstens 120 betragen. Je höher dieser Wert gewählt wird, desto länger dauert es bis die Hindernisse wieder von der Karte verschwinden. Bei der Verschmelzung (Abbildung 18) werden nur Werte berücksichtigt, die einen bestimmten Grenzwert (threshold siehe C.1) überschritten haben. Anschließend folgt die Normalisierung, bei der alle Werte, die größer als 1 sind, auf 1 gesetzt werden. Die so entstandene Karte kann nun an den Wavefront-Treiber übergeben werden, der nun mit Hilfe dieser dynamische Hindernisse umfahren kann.

Als besonderes Problem hat sich während der Implementierung des Treibers die Qualität der Lokalisationsdaten herausgestellt, die vom Treiber AMCL bereitgestellt werden. Der Algorithmus liefert nach einer anfänglichen manuellen Lokalisation Hypothesen für die aktuelle Position des Roboters und gibt die wahrscheinlichste aus. Diese Hypothese wirkt augenscheinlich gut und genügt auch, wenn man nur die ungefähre Position des Roboters haben möchte. Insbesondere der Treiber Wavefront verwendet diese Lokalisations-



(a) falscher Lokalisationswinkel (b) falsche Lokalisationsposition

Abbildung 19: Beispiele für eine falsch detektierte Position

werte und liefert zufriedenstellende Ergebnisse. Jedoch hat sich im Verlauf der Projektgruppe herausgestellt, dass die Werte nicht immer hinreichend gut mit der tatsächlichen Position übereinstimmen. Unser Treiber ist keine eigenständige Implementierung, sondern baut auf dem Playerframework auf und ist deshalb stark abhängig von der Qualität der implementierten Treiber.

Das Problem lässt sich hier in zwei Teilprobleme zerlegen. Zum einen können falsche Lokalisationswinkel ausgegeben werden. Die entstehenden Hindernisse in der Karte machen eine eigentlich mögliche Wegplanung unmöglich, da eine Tür fälschlicherweise als blockiert (Abbildung 19(a)) oder ein Gang als zu eng betrachtet wird. Zum anderen bereitet eine falsch lokalisierte X- und Y-Position Probleme. Entstehende Artefakte sieht man in Abbildung 19(b), wo die Wände versetzt eingezeichnet wurden, weil die X- und Y-Position leicht versetzt lokalisiert wurde. Wie bei den falschen Winkeln können auch hier Türen und enge Gänge irrtümlich als versperrt wahrgenommen werden.

Ein weiteres Problem ist die Asynchronität zwischen den Laserdaten und der Lokalisation. Mit Hilfe des Lasers wahrgenommene Hindernisse können so nicht ihrer exakten Position zugeordnet werden, was zur Folge hat, dass die Werte verschmieren.

Der Algorithmus lieferte in einer ersten stabilen Version zuverlässige Werte. Dies ist jedoch ausschließlich unter der Einschränkung möglich, dass nur Werte berücksichtigt wurden, wenn der Roboter sich bewegt und die Winkeländerung klein bleibt. Die oben dargestellten Probleme fallen hier kaum ins Gewicht, da der Roboter sich andauernd neu lokalisiert und die meisten Lokalisationen hinreichend genau sind. So verfallen fehlerhaft eingezeichnete Punkte direkt wieder und gelangen nicht in die Karte, die ausgegeben wird, da der Grenzwert (threshold siehe C.1) nicht überschritten wird. Ohne die oben erwähnten Einschränkungen würde der Algorithmus Ergebnisse wie in den Abbildungen 19(a) und 19(b) produzieren, sobald der Roboter stehen bleibt.

Natürlich ist es wünschenswert die Umgebung auch beobachten zu können, wenn der Roboter sich nicht mehr bewegt. Man stelle sich nur die Situation vor, dass der Roboter vor einer offenen Tür stehen bleibt. Die Tür wird jetzt geschlossen und der Roboter soll nun in einen Raum hinter dieser Tür fahren. Er plant durch die Tür, da der Treiber Wavefront das Hindernis noch nicht kennt und bemerkt, dass er sie nicht durchfahren kann. Er kann jedoch keinen Weg durch eine weiter entfernte zweite Tür planen, da der Treiber Wavefront immer den ersten Weg wählt.

Nachfolgend sollen die beiden Ansätze zur Lösung beschrieben werden. Zunächst einmal die Lösung für

falsche Lokalisationswinkel. Bei Betrachtung der Lokalisationswinkel sieht man, dass sie größtenteils gute Ergebnisse liefern. Unsere Idee war es, einen solchen guten Lokalisationswinkel erkennen zu können und anschließend auf diesem Wert mit den Werten der Odometrie weiterzurechnen. Eine solche gute Lokalisation kann man z.B. daran erkennen, dass viele Punkte mit Wänden der Umgebungskarte identifiziert werden. Wie viele Wände und dynamische Hindernisse vorhanden sind, kann von Szenario zu Szenario sehr verschieden sein. Also muss das Verfahren zur Erkennung entsprechend flexibel sein. In unserer Implementierung wird ein übernommener Lokalisationswert, andauernd mit den Ergebnissen der aktuellen Lokalisation verglichen. Ist diese deutlich besser, d.h. sie hat mehr erkannte Wände und weniger erkannte dynamische Hindernisse, so wird deren Wert übernommen und anschließend weiterverwendet. Dieser Austausch hat zudem den Vorteil, dass eventuell Abweichungen der Odometrie immer ausgeglichen werden. Die ursprüngliche Lokalisation wird bezüglich des Winkels stark verbessert. Das Problem kann jedoch nicht immer zufriedenstellend gelöst werden, da spezielle Konstellationen eine schlechte Lokalisation als gut erscheinen lassen können.

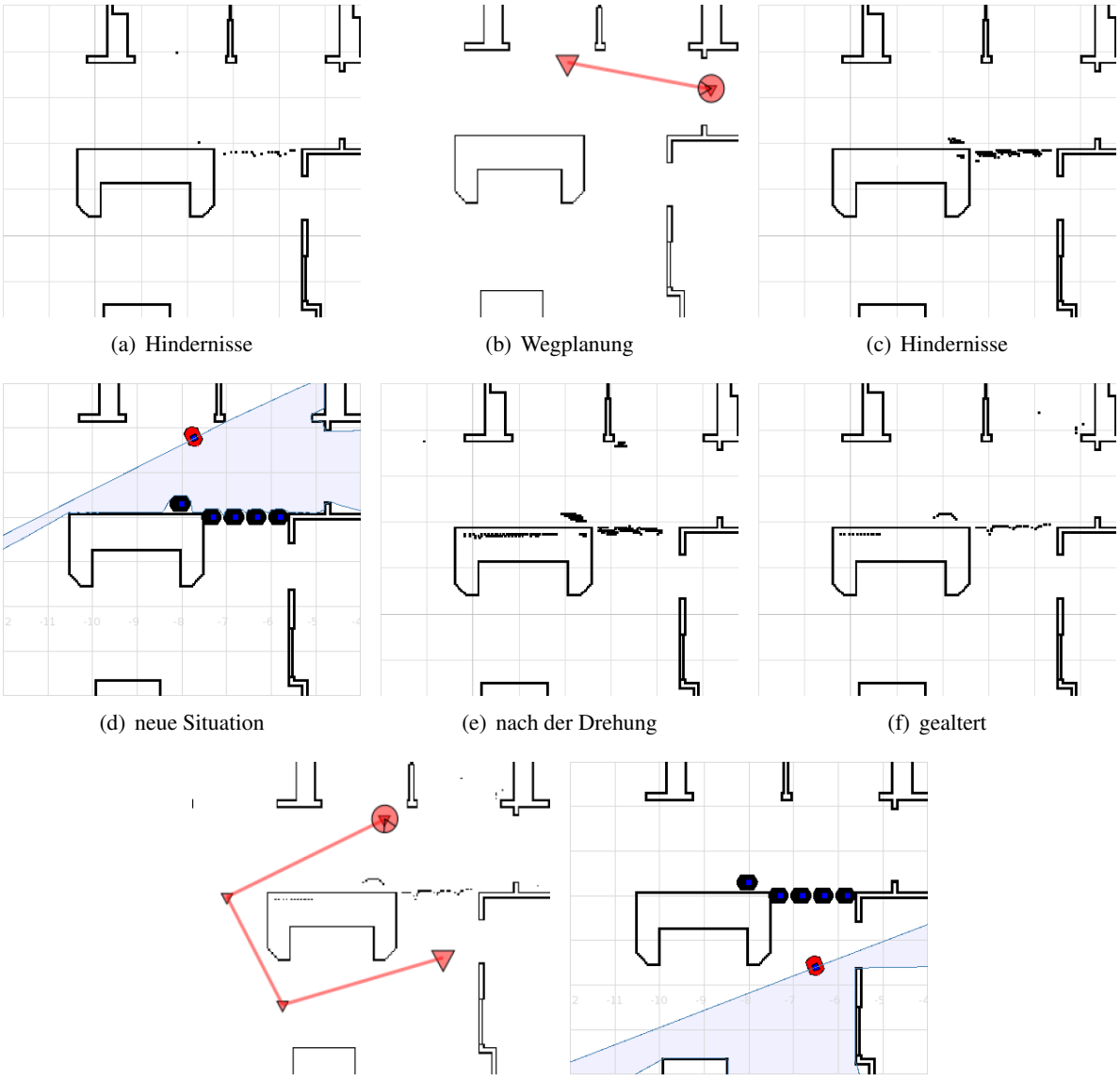
Um das Problem einer falschen Position bei der Lokalisierung zu lösen, wird nicht nur die aktuelle Position untersucht, sondern verschiedene Positionen um diese herum, wobei die X- und Y-Koordinaten verschoben sind (dislocation siehe C.1). Dabei wird überprüft, wie viele mögliche Wände an der jeweiligen Position erkannt werden. Es liegen nun Werte von verschiedenen Positionen vor. Jetzt wird überprüft in welchem Bereich die größten Werte liegen und wir nehmen an, dass die tatsächliche Position sich in der Nähe befindet. Die an dort angrenzenden Punkte werden bezüglich der Anzahl der möglichen Wände prozentual verglichen. Am Ende wird ein Punkt errechnet, der zu jenem mit der größten Anzahl tendiert, oder sogar auf diesem liegt, wenn der Wert die anderen dominiert. Dieser Punkt wird als aktuelle Position angenommen, um die Hindernisse auf der Karte einzuzeichnen. Diese Erweiterung zeigt Stärken und Schwächen. Zum einen werden kleine Verschiebungen sehr gut korrigiert, zum anderen können große Verschiebungen nicht korrigiert werden. Bei dynamischen Umgebungen können zusätzliche Probleme entstehen. Eine geöffnete Tür, die parallel zur dahinterliegenden Wand steht, kann diese Erweiterung z.B. negativ beeinflussen, da sie irrtümlich als Wand angesehen werden kann.

Die Funktionalität des Algorithmus, d.h. welche der gerade beschriebenen Einzelfunktionen benutzt werden, kann über den Parameter *functionality* gewählt werden (siehe C.1). Beide Erweiterungen können einzeln oder in Kombination genutzt werden. Ebenso kann die Funktionalität komplett ausgeschaltet werden.

In der Regel stehen Hindernisse, die eine Tür blockieren, in deren Mitte. Die Randbereiche der Türen können auf der Umgebungskarte so markiert werden, dass dort keine Hindernisse eingezeichnet werden. Die weiter oben erwähnten Fehler der Lokalisierung können so nicht mehr zur Blockierung der Tür führen. Dazu müssen die entsprechenden Pixel in der Umgebungskarte, die als 256-Farben-Bitmap repräsentiert wird, mit einem Farbwert zwischen 1 und 254 belegt werden.

Nachdem die Funktionsweise ausführlich beschrieben wurde, wird an dieser Stelle noch einmal auf das in Abbildung 15 vorgestellte Beispielszenario Bezug genommen. Die einzelnen Schritte im Ablauf des Szenarios sind in Abbildung 20 dargestellt. Die Simulation wurde mit der zum Playerframework gehörigen Simulationssoftware Stage durchgeführt.

Der Roboter erkennt zunächst die anderen Roboter (Abbildung 20(a)), die hier das Hindernis darstellen, und plant den Weg (Abbildung 20(b)). Er fährt los und nimmt die anderen Roboter immer noch wahr (Abbildung 20(c)). Durch die Bewegung verwischen die Positionen jedoch leicht. Der Roboter erreicht die Position B



(a) Hindernisse (b) Wegplanung (c) Hindernisse

(d) neue Situation (e) nach der Drehung (f) gealtert

(g) Umfahren (h) Fahrtziel erreicht

Abbildung 20: Einzelschritte des Beispielszenarios

und dreht sich bis er die Zielposition erreicht hat (Abbildung 20(d)). Abbildung 20(e) zeigt, dass der Roboter die anderen Roboter als Hindernisse auffasst. Jedoch hat er sich gerade noch bewegt und gedreht, was die Detektionen verwischt. Wartet man die Alterung ab, so nimmt der Roboter die Hindernisse schärfer wahr (Abbildung 20(f)). Wird nun der Weg zu Punkt C geplant, so sieht man, dass der Roboter den Weg außen wählt (Abbildung 20(g)). Am Ende erreicht er die Zielposition C (Abbildung 20(h)).

3.2.2 Integration in das Framework durch OSGi

Das OSGi-Bundle des Dynamicmapping-Treibers entscheidet darüber, ob ein aktueller Weg beibehalten oder ein neuer Weg geplant wird und zu welchem Zeitpunkt dies geschieht. Die Entscheidung ist so implementiert worden, um die Funktionen als „Service“ nutzen zu können.
Im Folgenden wird beschrieben, wie entschieden wird ob ein neuer Weg geplant werden soll.
Der Dynamicmapping-Treiber zeichnet dynamische Hindernisse auf die aktuelle Karte. Wenn nun eines dieser Hindernisse den aktuellen Weg des PARTYBOT blockiert, z.B. eine geschlossene Tür, bleibt der Roboter davor stehen. Nun muss die Entscheidung getroffen werden, ob er neu planen soll. Da der Roboter versucht, mit minimaler Abweichung von seinem Weg, um das Hindernis herumzufahren, bewegt sich der Roboter nur noch geringfügig auf der Stelle. Hierzu vergleichen wir die aktuellen mit vorherigen Positionswerten. Bei dem Roboter selbst kommt es hier zu einem „Wackeln“. Das bedeutet, der PARTYBOT versucht erst an der einen Seite vorbeizufahren, erfasst dann die Daten des Hindernisses neu und versucht dies dann auf der anderen Seite ebenso. Dieses „Wackeln“ des Roboters fangen wir ab, indem wir eine Winkelabfrage durchführen, in der kontrolliert wird, ob ein Vorzeichenwechsel vorliegt, d.h. ob der Roboter seine Drehrichtung geändert hat. Falls ein solches „Wackeln“ und damit ein Vorzeichenwechsel der Winkel vorliegt wurde das Hindernis erkannt und ein neuer Weg kann geplant werden. Im anderen Fall, wenn keine Vorzeichenänderung vorliegt, ist der PARTYBOT entweder manuell während seiner Route gestoppt worden oder er dreht sich um die eigene Achse in eine Richtung, um seine Route zu korrigieren.
Dies läuft solange, bis der Roboter entweder sein Endpunkt erreicht hat oder es keinen neuen Weg gibt auf dem er es erreichen kann. In diesem Fall setzt er sein Ziel auf seine aktuelle Position, um damit die Planung zu beenden. Es wird ein Rückgabewert von dem OSGi-Service, der diese dynamische Pfadplanung realisiert, geliefert. Dieser gibt Auskunft darüber, ob der Zielort erreicht wurde oder die Planung ohne Erfolg abgebrochen wurde.

3.3 Reaktive Navigation

Mobile Roboter, die in hoch dynamischen Umgebungen agieren, müssen ein reaktives Verhalten besitzen, um auf unvorhersehbare Ereignisse reagieren zu können. Sie sollten auf Personen oder allgemein auf bewegliche Objekte, die sich auf sie zubewegen, reagieren, indem sie zurückweichen Ein weiteres reaktives Verhalten besteht darin einer Person hinterher zu fahren, um zum Beispiel ein Foto von dieser an einer anderen Stelle zu erstellen. Ein wichtiger Aspekt bei der reaktiven Navigation ist, dass die Hindernisvermeidung aktiv bleibt und der Roboter nicht mit anderen Objekten kollidiert.

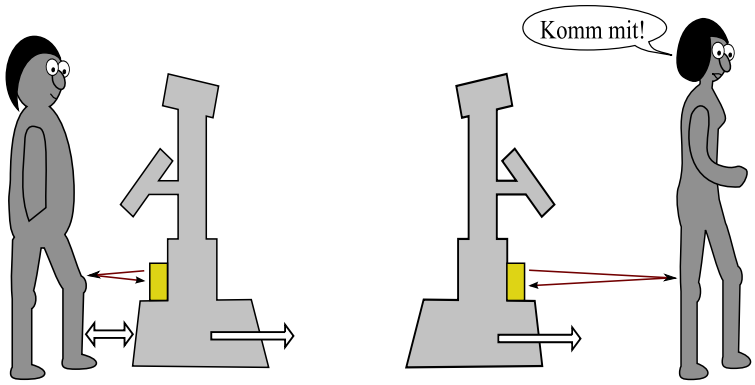


Abbildung 21: Reaktive Navigation: Ausweichen (links) und Verfolgen (rechts)

3.3.1 Lasergestützte Personenverfolgung

Um Personen zu verfolgen benutzt der PARTYBOT seinen Laser und detektiert mit diesem die Beine der Personen, die sich in Reichweite des Lasers aufhalten. Zunächst werden aus dem Laserscan Segmente, die ähnlich einem Bein strukturiert sind, gesucht und extrahiert. Anschließend wird die Beinbreite, sowie der Abstand der Beine überprüft, um Fehldetektionen zu verringern. Nun wird ein Beinpaar ausgewählt, welches verfolgt werden soll. In der nächsten Lasermessung wird versucht dieses Beinpaar wieder zu finden. Wenn dieses gefunden wird, versucht der Roboter die selektierten Beine vor sich zu zentrieren und einen festgelegten Abstand zu diesen einzuhalten (siehe Abbildung 21 rechts). Somit kann der Roboter Personen, die sich im Raum bewegen, verfolgen.

3.3.2 Angstverhalten

Im Rahmen der Projektgruppe wurde der PARTYBOT mit einem ängstlichen Verhalten versehen. Durch entsprechende äußere Einflüsse erfolgen Reaktionen wie einfaches Zurückweichen oder passende Audioausgaben. Der Lasersensor wird benutzt, um auf Personen reagieren zu können, die dem Roboter zu nahe kommen. Dazu werden die beiden Mikrofone des Roboters als Sensoren verwendet, um laute Geräusche zu detektieren. Über die Mikrofone lassen sich ebenfalls Berührungen des Robotergehäuses erkennen, da diese durch den Körperschall weitergegeben werden. Im Angstzustand weicht der PARTYBOT zurück, wenn sich jemand bis auf wenige Zentimeter nähert (siehe Abbildung 21 links). Er gibt einen Hilferuf über die Lautsprecher aus, wenn laute Geräusche in der Nähe der Mikrofone erzeugt werden oder sein Gehäuse unsanft berührt wird.

4 Steuerarchitektur

Um die Handlungen des Roboters nach einem festgelegten Szenario (siehe Abschnitt 1.2) zu ermöglichen, bedarf es dem entsprechenden Steuerungsmechanismus, die sogenannte *Steuerarchitektur*. Für diesen Begriff existieren verschiedene Definitionen, die Intentionen des jeweiligen Steuerungskonzeptes zum Ausdruck bringen sollen. In dieser Arbeit definieren wir eine Steuerarchitektur als Struktur der Steuerung eines Robotersystems. Etwas abstrakter betrachtet, kann man sich ein Robotersystem als eine Blackbox vorstellen:

$$x \rightarrow \boxed{\text{Black-Box}} \rightarrow y$$

Abbildung 22: Prozessdatenverarbeitung [57]

Eingänge der Blackbox sind die Sensormesswerte, die durch Funktionen und Algorithmen in die physikalischen Reaktionen der Aktoren umgewandelt werden. Die Blackbox bzw. deren Struktur kann man somit als eine Steuerarchitektur ansehen.
Ausgehend von der Definition befassen wir uns in diesem Kapitel zuerst mit der Zielsetzung einer Steuerarchitektur und sich daraus ergebenden Anforderungen und untersuchen die Taxonomie der Steuerarchitekturen, um ein geeignetes Konzept für den Entwurf zu motivieren. Im Kapitel 4.2 wird das von uns entwickelte Konzept der *Aufmerksamkeitsbasierten Steuerarchitektur* vorgestellt. Für die Umsetzung dieses Konzeptes haben wir das *OSGi-Framework* dafür benutzt, um insbesondere Modularität (siehe Anforderungen in 4.1) der Steuerarchitektur durch die Implementierung einer *serviceorientierten Architektur (SOA)* zu gewährleisten. Im Kapitel 4.3.1 werden OSGi-Framework sowie das SOA-Konzept erläutert. Anschließend wird im Kapitel 4.4 die technische Realisierung des entworfenen Konzeptes in OSGi beschrieben. Insbesondere wird dabei auf die implementierten Funktionalitäten bzw. Dienste eingegangen.

4.1 Steuerarchitekturen für mobile Roboter

Eine Steuerarchitektur verfolgt mehrere Ziele. Sie soll an erster Stelle die Implementierung und Wartung von komplexen Anwendungen erleichtern und dabei helfen, typische Modellierungsfehler von vornherein zu vermeiden. Um diese Ziele zu erfüllen, existiert eine Reihe von Anforderungen, die dafür in einer Steuerarchitektur umgesetzt werden sollen. Nach [2] ergeben sich insbesondere

- Effektivität in Hinsicht der Erfüllung der Aufgabe
- Unterstützung für Parallelisierung, d.h. die Verfolgung mehrerer Ziele zur gleichen Zeit. Das System muss auch damit fertig werden, dass die Wichtigkeit der Ziele von der aktuellen Situation abhängig sein kann.
- Robustheit, d.h. die Fähigkeit, mit unerwarteten Eingaben zurechtzukommen. Unerwartetete Sensoreingaben können durch unerwartete Umweltereignisse auftreten. Wenn ein oder mehrere Sensoren ausfallen, soll das System trotzdem weiterhin funktionieren und nicht komplett ausfallen.
- Mehrsensorfähigkeit: Sensoren sind fehlerbehaftet, deswegen muss die Systemarchitektur, ausgehend von gelieferten Sensordaten, die richtige Entscheidung für das weitere Handeln treffen.

- Erweiterbarkeit: Die Steuerarchitektur ist umso besser, je weniger Aufwand betrieben werden muss, um neue Funktionalitäten in ein bereits bestehendes System einzubinden.
- Modularität
- Flexibilität zur Laufzeit

Nachdem die Anforderungen an eine Steuerarchitektur festgelegt sind, muss nun die Architektur präzise definiert bzw. beschrieben werden. Eine mögliche Beschreibung wurde in [37] vorgeschlagen und beruht auf der Definition und Abhängigkeiten der Komponenten *SENSE*, *PLAN* und *ACT*.

SENSE steht für das Erfassen und Verarbeiten von Sensordaten.

PLAN verknüpft die eingelesenen Daten mit schon vorhandenen Daten und ist für die Ermittlung eines Plans, einer Vorgehensweise, zuständig.

ACT beeinflusst die Steuerung der Aktoren und damit die Bewegungen des Roboters.

Das Zusammenspiel dieser Komponenten charakterisiert eine Steuerarchitektur, wobei deren Auswahl optional sein kann, z.B. in *reaktiven* Systemen entfällt die Planungskomponente. Ausgehend von der Auswahl der Komponenten existieren reaktive (siehe Abschnitt 4.1.1), *deliberative* (siehe Abschnitt 4.1.2) und *hybride* (siehe Abschnitt 4.1.3) Steuerarchitekturen, wobei hybride Ansätze eine Kombination von reaktiven und deliberativen Ansätzen darstellen. Eine weitere Art von Steuerarchitekturen bilden die *verhaltensbasierten* Ansätze (siehe Abschnitt 4.1.4), die nach ihren Merkmalen zwischen reaktiven und deliberativen Steuerarchitekturen angeordnet werden können, in der Klassifikation aber zu den reaktiven Ansätzen gehören.

4.1.1 Reaktive Steuerarchitekturen

Reaktive Systeme können durch die Komponenten SENSE und ACT beschrieben werden. Dabei verbinden sie direkt die Sensoreingaben mit den Aktorausgaben.

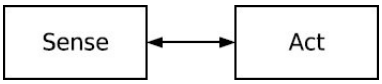


Abbildung 23: Reaktiver Ansatz

Durch das Verzichten auf die Planungskomponente ist es möglich, ein schnelles Antwortverhalten in dynamischen Umgebungen zu gewährleisten. Hintergrund dabei ist der Bezug auf das *Stimulus-Response-Schema* der Biologie. Der Verzicht auf die Planungskomponente, und damit auf ein *Weltmodell*, macht es für solche Systeme unmöglich, sich in einer lernenden Art auf seine Umgebung einzustellen. Rein reaktive Systeme eignen sich also für schnelle Reaktionen in Umgebungen, die zur Entwurfszeit bekannt sind. Das Kredo von reaktiven Systemen ist: „Denke nicht! Handle!“ [36]

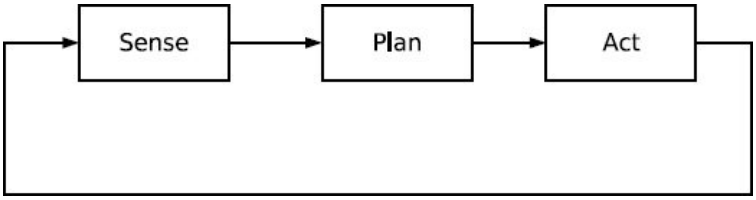


Abbildung 24: Deliberativer Ansatz

4.1.2 Deliberative Steuerarchitekturen

Deliberative Systeme verwenden alle drei oben beschriebenen Komponenten. Aus den eingelesenen Sensordaten und einem internen Zustand wird ein Weltmodell generiert, das als Eingabe für die Planungskomponente verwendet wird.
Die Planungskomponente erlaubt es, auch komplexere Aktionen, ausgehend von einem Weltmodell, zu berechnen. Nachteil solcher Systeme ist es, dass als Folge der Änderung in der Umgebung eine Neuberechnung des Weltmodells stattfindet und daher solche Systeme langsam arbeiten. Das Kredo bei deliberativen Systemen heißt: „Erst genau nachdenken! Dann handeln!“ [36]

4.1.3 Hybride Steuerarchitekturen

Hybride Systeme haben das Ziel, die Vorteile von reaktiven und deliberativen Systemen zu nutzen, also die Echtzeitfähigkeit der reaktiven Systeme mit der Rationalität der deliberativen zu verknüpfen.

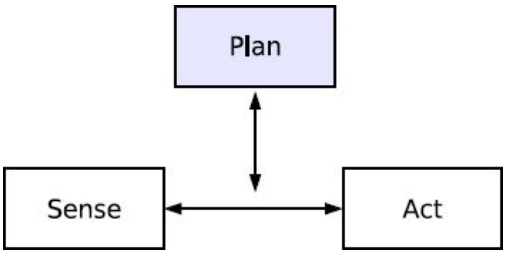


Abbildung 25: Hybrider Ansatz

Dabei kann sich der reaktive Teil eines solchen Systems um die einfachen Handlungen, wie z.B. Kollisionsvermeidung kümmern, während der deliberative Teil für komplexere Aufgaben, wie z.B. Zielverfolgung benutzt wird. Da beide Komponenten unterschiedliche Aufgaben erfüllen, ist die Koordinierung der Komponenten nicht einfach und muss von einer weiteren Komponente erledigt werden. Das Kredo einer solchen Steuerung lautet: „Denke und handle unabhängig und gleichzeitig!“[36]

4.1.4 Verhaltensbasierte Steuerarchitekturen

Verhaltensbasierte Steuerarchitekturen werden als Erweiterung von reaktiven Ansätzen angesehen. Diese Architekturen beruhen zwar auf reaktiven Ansätzen, besitzen aber Fähigkeiten, die nicht nur auf direkte Verknüpfung von Eingabewerten mit den Ausgabewerten beschränkt sind. Grundlage dafür ist der Begriff *Verhalten*. In der Literatur ist dieser Begriff nicht eindeutig definiert. Für die vorliegende Arbeit sind zwei Definitionen von Verhalten relevant, die im Folgenden beschrieben werden.

In [57] wie auch in den meisten Arbeiten zu verhaltensbasierten Steuerarchitekturen wird Verhalten als „ein einfaches Modul, welches gelesene Eingangsdaten bzw. Sensordaten mit Algorithmen und Funktionen auf Ausgangsdaten bzw. Aktionen abbildet“, definiert. Wenn ein solches Modul direkt, sehr schnell und unabhängig vom internen Zustand auf bestimmte Ereignisse (Eingangsdaten) reagiert, wird Verhalten hinsichtlich der biologischen Definition als *Reflex* definiert.
Eine andere Definition vom Verhalten gibt Steels [56]. Dort wird Verhalten als eine „Abfolge von beobachteten und interpretierten Aktionen eines Roboters in seiner Umgebung“ definiert. In dieser Definition wird Verhalten ähnlich dem Begriff aus der Verhaltens-Psychologie charakterisiert. Somit ist Verhalten kein spezielles Modul, sondern das Zusammenspiel von verschiedenen reaktiven Modulen mit der Umwelt.
Weiterhin wurden für verhaltensbasierte Steuerarchitekturen folgende Annahmen nach Rodney A. Brooks (vgl. [8]), die solche Ansätze charakterisieren, gemacht:

Situationsbezogenheit Die Welt wird nicht durch ein Modell ersetzt, sondern die Welt selbst ist ihre beste Repräsentation.

Verkörperung Der Roboter soll in seiner Umgebung eine physikalische Präsenz haben, d.h. er muss seine Umwelt aktiv betrachten und beeinflussen können.

Intelligenz Es soll keine klare Trennlinie zwischen dem intelligenten und reaktiven Verhalten existieren.

Emergenz Intelligenz wird durch die Umwelt bestimmt, mit der der Roboter interagiert.

Bei der Betrachtung verschiedener verhaltensbasierter Architekturen stellt sich heraus, dass der Hauptunterschied der verschiedenen Steuerarchitekturen in der Methode der Koordination von Verhalten besteht. Eine mögliche Klassifikation der Verfahren zur Verhaltenskoordination wurde von Paolo Pirjanian [46] vorgeschlagen. Dabei werden in einem System auf der obersten Ebene entweder die einzelnen Verhalten ausgewählt (*Verhaltensselektion*) oder sie laufen parallel.

Verhaltensselektion

Koordinierung der einzelnen Verhalten wird dabei durch die Auswahl eines aktiven Verhaltens realisiert. Wenn ein Verhalten ausgewählt wurde, darf dieses direkt auf die entsprechenden Aktoren zugreifen. Eine solche Verhaltensselektion kann entweder *zentral*, d.h. durch einen übergeordneten Mechanismus oder *dezentral*, d.h. durch verteilte Verfahren realisiert werden. Die Einschränkung solcher Ansätze besteht darin, dass die einzelnen Verhaltensmuster nur sequentiell, also nacheinander, aktiviert werden können. Im Falle eines Systems mit zwei Verhalten, „Bewegung zum Ziel“ und „Hindernisvermeidung“, kann somit keine optimale Lösung gefunden werden, da die Berechnungen der einzelnen Verhalten nacheinander durchgeführt werden. Beispiele für solche Ansätze sind die *Subsumption-Architektur* von Brooks und *Activation Networks* von Maes [2].

Parallel laufende Verhalten

Parallel laufende Verhalten bilden eine Alternative zur Verhaltensselektion. Hierbei werden die Ausgaben einzelner Verhaltensweisen zu einer resultierenden Aktion verschmolzen. So ist es möglich mehrere Ver-

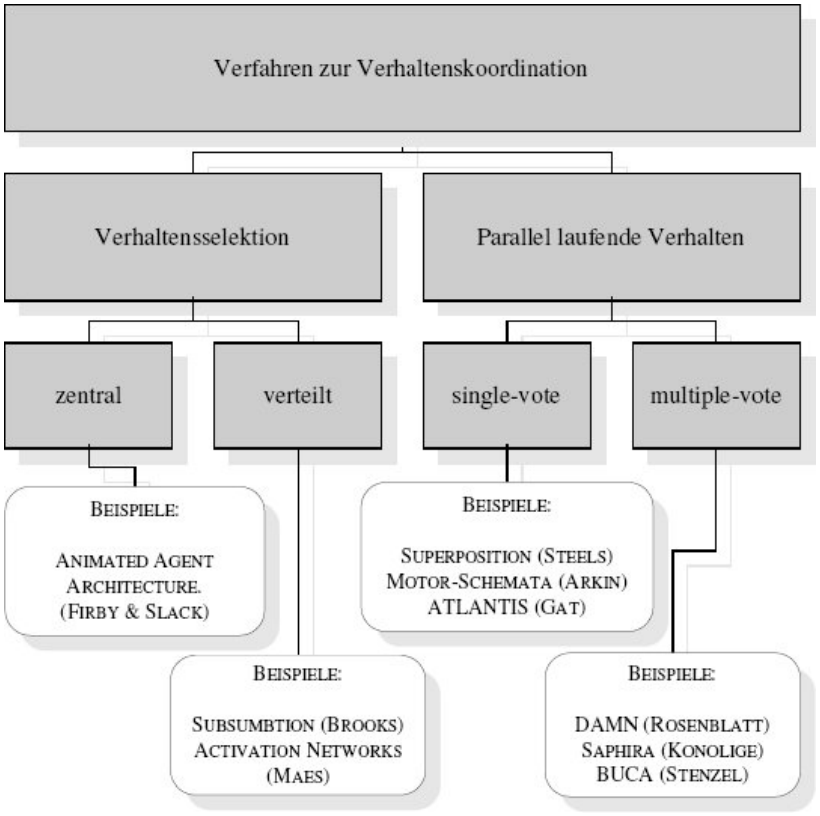


Abbildung 26: Klassifikation der Koordinationsverfahren für verhaltensbasierte Steuerarchitekturen [56]

halten nebenläufig zueinander zu benutzen. Die Koordinierung der einzelnen Verhaltensausgaben stellt hier das Hauptproblem dar.
Die Koordination einzelner Verhalten kann auch als Abstimmungsprozess aufgefasst werden. Jedes Verhalten gibt seine Stimme in einer geeigneten Form ab, und nach einer Abstimmungsregel wird die Ergebnisaktion bestimmt, die ausgeführt werden soll. Die parallel laufenden Verhaltensweisen unterteilen sich in die *Single-Vote-* und *Multiple-Vote-*Verfahren.
Bei den Single-Vote-Verfahren ist es nur erlaubt, dass jedes Verhalten für genau eine Aktion abstimmt. Dabei ist es also nicht möglich, dass die Verhalten für mehrere Aktionen oder nur für eine Aktion abstimmen. Beispiel für einen solchen Ansatz ist die *Motor-Schemata* von Arkin [2].
Multiple-Vote-Verfahren kann man als ein Bewertungs- oder Abstimmungsprozess ansehen, in dem jedes Verhalten für mehrere Aktionen abstimmen kann. Dadurch wird es für jedes Verhalten möglich, alle möglichen Aktionen eines Aktors zu bewerten. Die resultierende Aktion stellt somit einen Kompromiss bzgl. der Präferenzen der verschiedenen Verhalten dar. Beispiel für einen solchen Ansatz ist die *Distributed Architectur for Mobile Navigation* (DAMN) von Rosenblatt [2].

4.1.5 Resumee

Ausgehend von der oben beschriebenen Taxonomie der Steuerarchitekturen stehen für die Erfüllung der Aufgabenstellung in dieser Arbeit an erster Stelle die verhaltensbasierten Ansätze. Wie es schon oben erläutert wurde, stellt der Begriff Verhalten die Grundlage für den Entwurf solcher Architekturen. Wei-

terhin sollen bei einem verhaltensbasierten Ansatz eine Reihe von *Verhaltensmodulen* geeignet koordiniert werden. Ein solcher Koordinationsmechanismus bildet die Grundlage für die Entscheidung des Systems für ein weiteres Vorgehen und somit auch die Grundlage für die Aufmerksamkeit.
Da die verhaltensbasierten Ansätze nicht rein reaktive Ansätze sind, sondern (in der Klassifikation) zwischen den reaktiven und deliberativen Ansätzen angesiedelt sind, können mit solchen Steuerarchitekturen auch komplexe Handlungen realisiert werden, indem die Welt intern repräsentiert wird. Der Unterschied zu rein deliberativen Ansätzen besteht darin, dass die Welt selbst als die Grundlage für die Repräsentation angenommen wird und daher keine anderen Modelle der Welt generiert werden.

4.2 Aufmerksamkeitsbasierte Steuerarchitektur

Als erster Schritt zur Erstellung einer aufmerksamkeitsbasierten Steuerarchitektur wurden Anwendungsfälle (Use Cases) des Roboters zusammengetragen. Als Grundlage diente das Szenario einer Party auf welcher sich der Roboter frei bewegen soll und eine einfaches soziales Verhalten aufweist. Der Roboter soll also in einer Gruppe von Personen autonom navigieren und dort mit Personen Kontakt aufnehmen und Dienste anbieten so wie ggf. Anweisungen entgegen nehmen.

4.2.1 Use Cases

Es wurden folgende Anwendungsfälle definiert:

Person suchen Der Roboter sucht in dem Raum, in dem er sich befindet, nach Personen, dazu fährt er den Raum ab und versucht über die Personendetektion diese zu finden. Werden mehrere Personen erkannt, wählt der Roboter aus den möglichen Zielen das Interessanteste.

Zu Person fahren Wurde eine Person gefunden und als Ziel gewählt, navigiert der Roboter zu dieser Person. Dabei muss mehrfach überprüft werden, ob sich die Person noch an ihrer Ausgangsposition befindet oder sich von dieser wegbewegt hat.

Kontakt aufnehmen Hat der Roboter eine Person erreicht, nimmt er durch eine entsprechende Audioausgabe Kontakt auf. Über den integrierten Monitor werden dem Nutzer Dienste angeboten.

Foto machen Über das Touchpad kann der Nutzer mit Hilfe der Kamera ein Foto von sich machen und dieses per E-Mail versenden.

Person verfolgen Über das Touchpad kann der Nutzer dem Roboter die Anweisung geben ihn zu verfolgen. Dies kann jederzeit, ebenfalls über das Touchpad, wieder unterbunden werden.

Angst zeigen Wird der Roboter bedroht, reagiert er auf Personen und laute Geräusche mit defensiven Verhalten und entsprechenden Sprachausgaben. Eine Bedrohnung ist z.B. das schnelle Herantreten einer Person an den Roboter.

Pause Um Einstellungen am Roboter vorzunehmen, ist es sinnvoll, den Roboter anzuhalten. Daher ist es möglich, den Roboter per Knopfdruck aus allen anderen Zuständen zu pausieren.

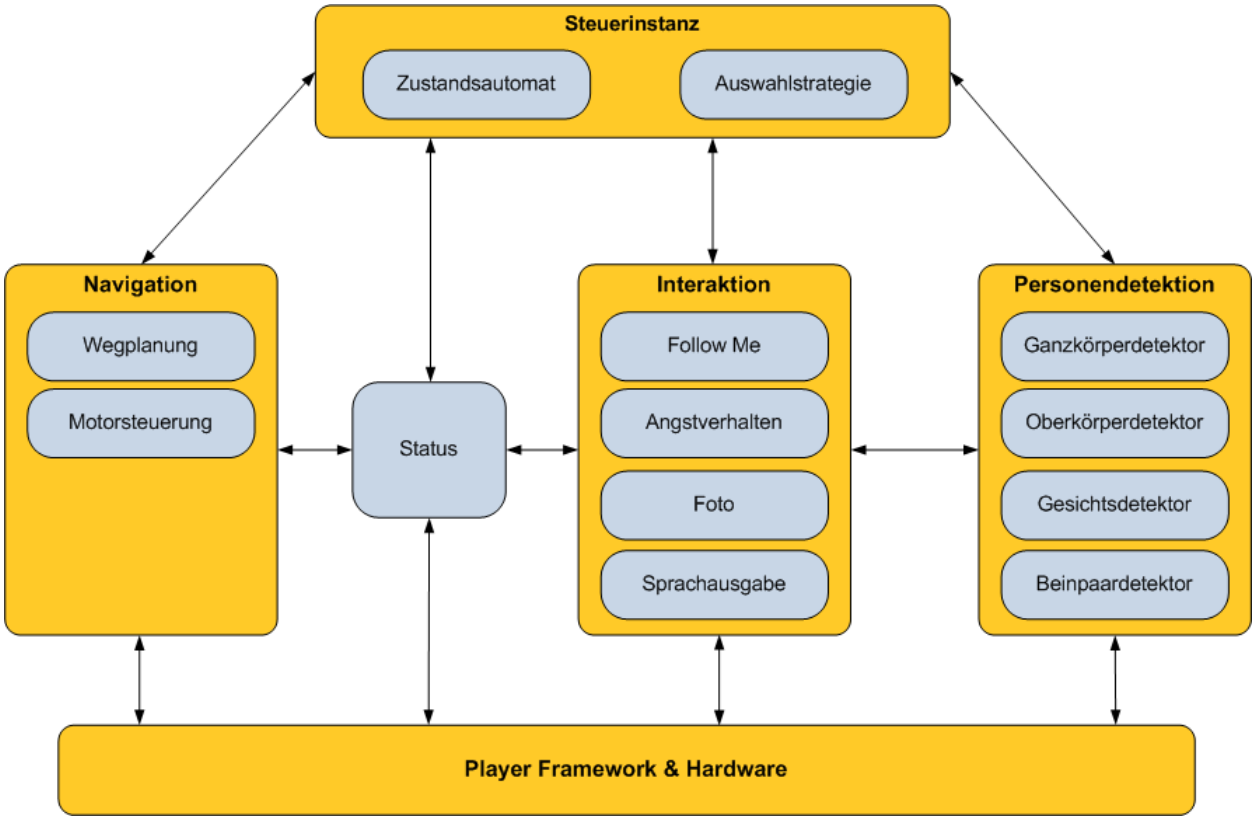


Abbildung 27: Schematischer Aufbau der Gesamtarchitektur

4.2.2 Aufbau der Gesamtarchitektur

Der Gesamtaufbau des Systems lässt sich in vier Teilbereiche unterteilen (siehe Abbildung 27). Es gibt eine Steuerinstanz, jeweils ein Teilsystem, das für die Navigation bzw. Personendetektion zuständig ist, und einen Bereich, der alle Funktionen mit denen der Roboter direkt mit seiner Umwelt interagiert, beinhaltet. Des Weiteren ist es möglich über einen Statusservice alle wichtigen Daten des Roboters auszulesen.
Das Grundlegende Prinzip der Architektur ist folgendes: Die Steuerinstanz nimmt mit Hilfe der Personendetektoren die Umwelt des Roboters wahr und entscheidet auf Basis dieser Daten den nächsten Schritt des Systems. Wird von den Personendetektoren eine oder mehrere potenzielle Personen erkannt, wird durch die implementierte Auswahlstrategie eine dieser Personen als Ziel ausgewählt und über die Navigation angesteuert. Ist das Ziel erreicht, kann der Roboter mit seiner Umgebung bzw. mit der Person interagieren.
Die Kommunikation zwischen den einzelnen Modulen ist über Services realisiert, die als Schnittstellen für die jeweiligen Funktionen dienen.

4.2.3 Zustandsautomat

Als zentrale Steuerinstanz wurde ein Zustandsautomat (Abbildung 28), welcher auf Events bzw. Eingaben von anderen Instanzen des Systems reagiert und den entsprechenden Zustandswechsel ausführt, entwickelt. Die Zustände des Automats wurden direkt aus den Use Cases abgeleitet. Es gibt für jeden Use Case einen Zustand, in dem sich der Roboter befinden kann. Bei der Implementierung wurde darauf geachtet, dass sich

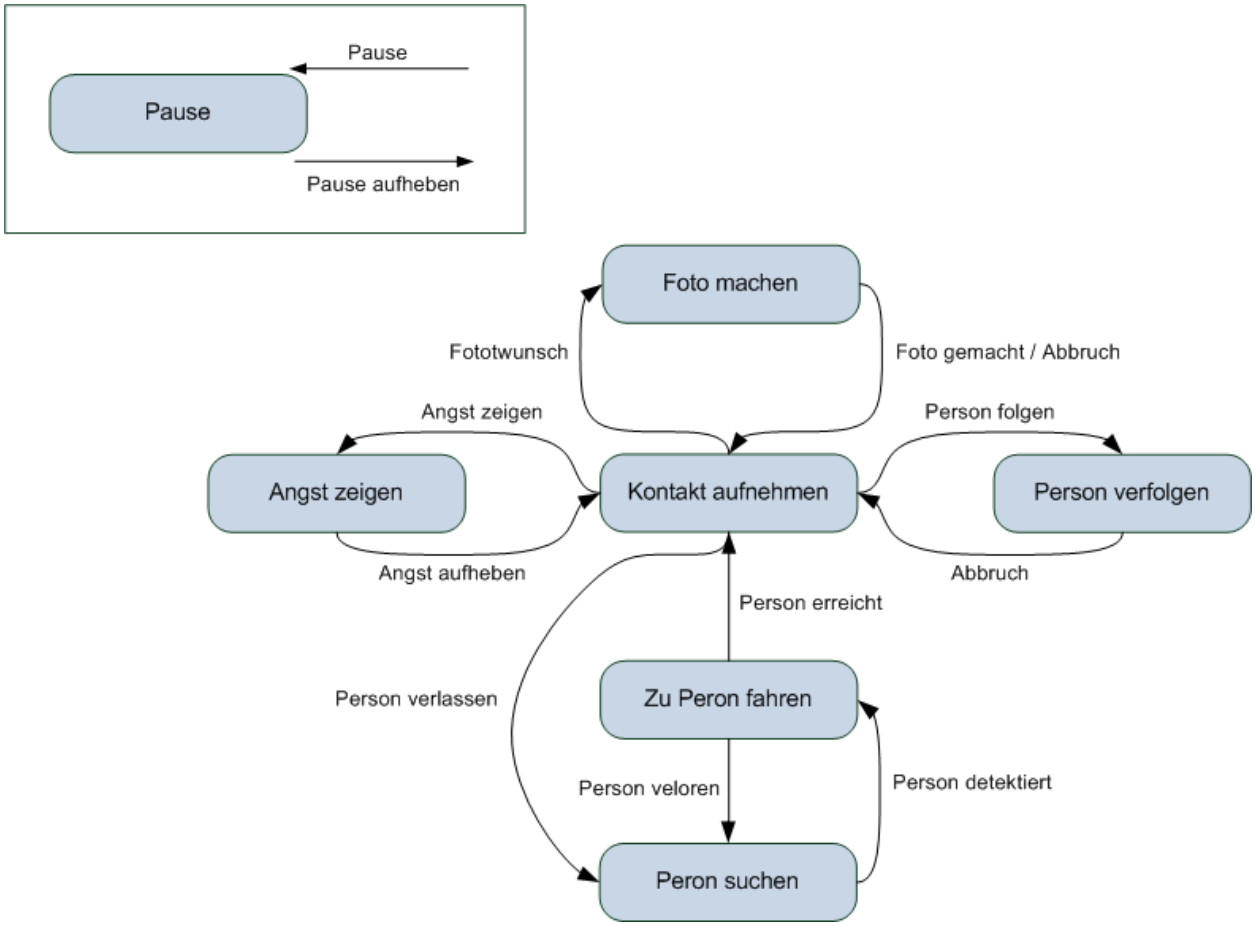


Abbildung 28: Eventgetriebener Zustandsautomat als zentrale Steuerinstanz

der Automat leicht um zusätzliche Zustände und Events erweitern lässt. Durch diesen dynamischen Ansatz ist es gewährleistet, dass sich der Roboter auch in größeren bzw. anderen Szenarien einsetzen lässt.

4.3 Umsetzung in einer serviceorientierten Architektur

Ein Ziel der Projektgruppe war es, die gesamte Architektur serviceorientiert zu modellieren. Die Tatsache, dass der Roboter aus vielen einzelnen erst einmal von einander unabhängigen Teilsystemen wie z.B Laser, Kamera oder Antrieb besteht, lässt sich sehr gut in eine solche serviceorientierte Architektur übertragen. Ein Framework, welches den Aufbau von serviceorientierten Architekturen unterstützt, ist OSGi.

4.3.1 OSGi

Das OSGi (Open Service Gateway initiative) Framework wird von der OSGi Alliance [42] spezifiziert und realisiert eine Serviceorientierte Architektur. Mitglieder der OSGi Alliance sind unter anderem Firmen, wie z.B. IBM, Oracle und Sun Microsystem. Mit dem OSGi Framework wird versucht, eines der größten Probleme, vor dem die heutige Softwareentwicklung steht, zu lösen: Die ständig anwachsende Komplexität der Programme. Die Idee hinter dem OSGi Framework ist es, einzelne Softwaremodule, sogenannte Bundles, weitesgehend unabhängig voneinander ausführen zu lassen. Eine Besonderheit hierbei ist es, dass

Bundles zur Laufzeit dynamisch in das Framework geladen und ausgetauscht werden können, ohne dass die restlichen Module oder das Framework selber davon beeinflusst werden oder gar neu gestartet werden müssen. Innerhalb des Frameworks kommunizieren die Bundles über Services. Die Implementierung des Frameworks basiert auf Java, dadurch bleibt die Portabilität einer Standard Java Applikation erhalten. Die Realisierung der Spezifikation der OSGi Alliance wird durch Dritte Implementiert, hier gibt es sowohl Open Source als auch kommerzielle Lösungen. Eine breite Anwendung findet OSGi als Framework in eingebetteten Systemen wie Fahrzeugen, Handys und bei der Heimvernetzung.

4.3.2 SOA

Wie schon angesprochen, entspricht das OSGi Framework der Realisierung einer serviceorientierten Architektur (siehe Abbildung 29). Hierbei wird die Programmlogik auf Services aufgeteilt. Jeder dieser Services ist in sich abgeschlossen und kann eigenständig genutzt werden. Jeder Service veröffentlicht innerhalb der Architektur seine Schnittstelle. Um einen Service zu nutzen, reicht es aus diese Schnittstelle zu kennen. Der große Vorteil liegt in der Modularität und der Wiederverwendbarkeit. Eine SOA kann grob in drei Bereiche aufgeteilt werden. Einmal die ServiceRegistry, in der sich jeder Service registrieren kann. Die Serviceprovider sind Module in der Architektur, die Services registrieren und so zur Verfügung stellen. Zuletzt gibt es noch Serviceconsumer, welche die in der Serviceregistry hinterlegten Services anfragen und nutzen.

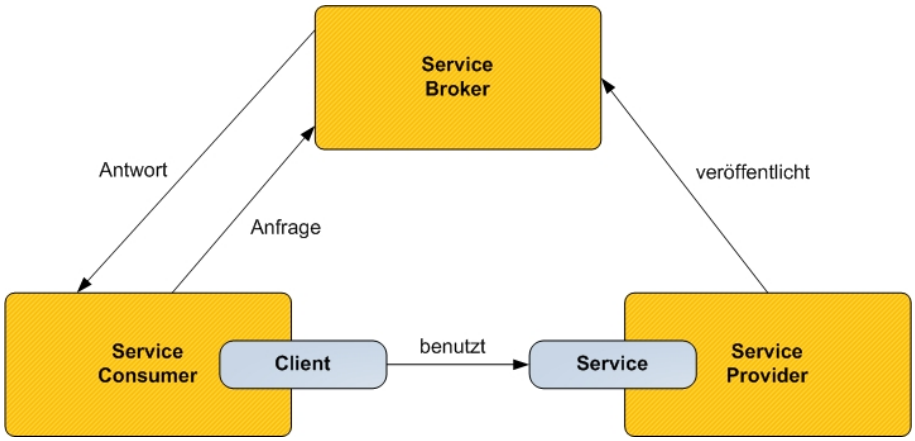


Abbildung 29: Aufbau einer serviceorientierten Architektur nach [42]

4.3.3 Funktionsweise des Frameworks

Die einzelnen Module in einem OSGi Frameworks heißen Bundles. Jedes Bundle kann eigene Services registrieren und auch Services von anderen Bundles nutzen. Die Granularität der Bundles ist dabei beliebig und kann von einer atomaren Funktionalität bis zu einem kompletten Programm reichen. Aus struktureller Sicht kann man das OSGi Framework in vier Schichten aufteilen:

Security Layer Der Security Layer in OSGi ist optional und basiert auf der Java 2 Security Architektur. Hier können z.B. die Rechte für Bundles und deren Authentifizierung durch Signaturen umgesetzt werden. Das Framework bietet hierzu Standardservices wie UserAdmin und PermissonAdmin bereit.

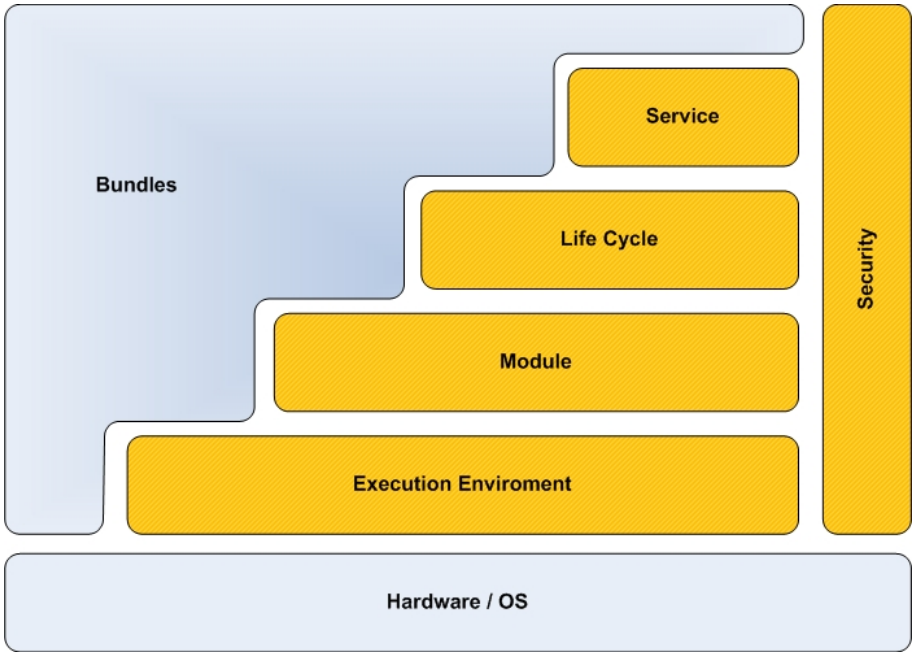


Abbildung 30: OSGi Layermodell nach [42]

Module Layer In dem Module Layer sind die Bundles angesiedelt. Das Framework selber ist ebenfalls als Bundle realisiert, dem sogenannten System Bundle. Auch werden durch die Spezifikation bereits eine Reihe von Standardbundles bereitgestellt, darunter fallen z.B. ein HTTP-Server, ein Log Service und ein Telnet Service. Außerdem wird in dem Module Layer der Austausch von Packages zwischen den Bundles organisiert. In den Packages werden insbesondere die Schnittstellen für die von den Bundles bereitgestellten Services weitergegeben und auch Abhängigkeiten zwischen den Modulen verarbeitet.

Life Cycle Layer Jedes Bundle im Framework durchläuft einen Lifecycle. Dabei kann es die Zustände *Installed*, *Resolved*, *Starting*, *Active*, *Stopping* oder *Uninstalled* annehmen. Im Zustand Installed ist das Bundle in das Framework eingehängt, bietet aber keinerlei Services oder sonstige Funktionalität. Aus diesem Zustand geht das Bundle automatisch in den Zustand Resolved über. Dieser Zustand wird auch eingenommen, wenn ein aktives Bundle gestoppt wird. Ist ein Bundle Resolved, sind alle benötigten Javaklassen verfügbar und das Bundle ist startbereit. Wird der Startvorgang über den Zustand Starting erfolgreich ausgeführt, ist das Bundle im Zustand Active. Jetzt läuft das Bundle und führt seine Funktionalität aus und kann Services nutzen oder selber zur Verfügung stellen. Wird ein Bundle gestoppt, geht es in den Zustand Stopping. Wird dieser erfolgreich durchlaufen, befindet es sich wieder im Zustand Resolved. Möchte man ein Bundle wieder aus dem Framework entfernen, muss man es deinstallieren und es erreicht den Endzustand Uninstalled.

Service Layer Der Kern jeder serviceorientierten Architektur sind Services. Technisch gesehen sind im OSGi Framework Services Instanzen von Java-Objekten, die die Methoden einer Inferfaceklasse implementieren. Diese Interfaces wird anderen Bundles per Package Export / Import bereitgestellt. Über diese Interfaces können dann die Methoden des Serviceobjektes genutzt werden.

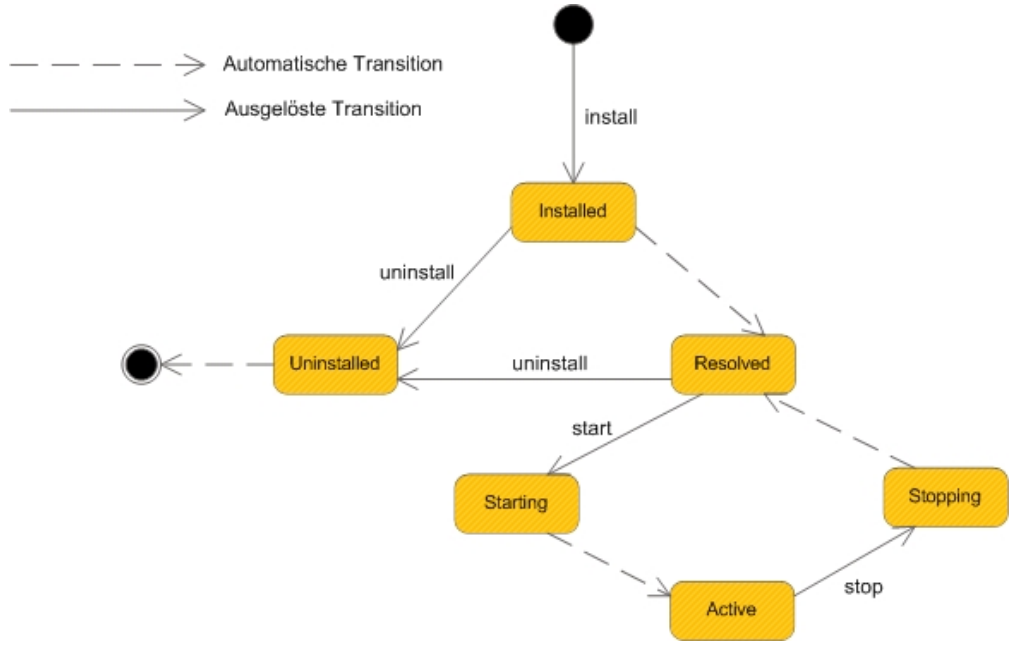


Abbildung 31: OSGi Lifecycle nach [42]

4.4 Implementierung mit OSGi

Ein Bundle ist eine jar-Datei, die eine Activator.class-Datei und eine MANIFEST.TXT-Datei beinhaltet. Wie schon erwähnt, sind Services Javaklassen, die ein Interface implementieren. Die Activator.class entspricht im Prinzip der Main.class eines normalen Javaprogramms. Durch die Implementierung des Interface BundleActivator enthält die Klasse zwei Methoden. Einmal die start-Methode, welche ausgeführt wird, wenn sich das Bundle im Starting Zustand befindet. Die Methode ist der Einstiegspunkt des Bundles und entspricht der main-Methode eines Standard-Javaprogramms. Die andere Methode, die durch das Interface implementiert wird, ist die stop-Methode. Diese wird aufgerufen, wenn ein Bundle gestoppt wird. Hier müssen insbesondere laufende Threads beendet werden, da diese sonst im Hintergrund weiterlaufen. Im Manifest werden Metainformationen für das entsprechende Bundle bereitgestellt. Notwendigerweise muss hier der Name und Pfad der Activatorklasse innerhalb der jar-Datei angegeben werden. Des Weiteren wird hier der Import und Export von Packages, der Bundlename, Versionsinformationen und vieles mehr festgelegt. Stellt ein Bundle Services bereit, sind diese Klassen ebenfalls im Bundle enthalten. Wird ein Service von einem Bundle im Framework registriert, kann dieser zusätzlich mit Properties ausgestattet werden. Dadurch können andere Bundles es per Suchanfrage in der Service Registry einfacher finden. Will ein Bundle ein Service nutzen, erhält es vom Framework die entsprechende ServiceReference, über die der Service genutzt werden kann. Will ein Bundle mit dem Framework interagieren, wird dies über den BundleContext, welcher als Parameter der start-Methode mitgegeben wird, realisiert. Über den BundleContext kann ein Bundle Services registrieren und Services anderer Bundles suchen und nutzen. Wird ein Service im Framework registriert, löst es das Event REGISTERING aus. Ein Service kann noch zwei weitere Events auslösen. Das UNREGISTERING Event, wenn der Service wieder aus dem Framework entfernt wird, und das MODIFIED Event, wenn ein Service erneuert wird. Auf diese Events, kann entweder mit einem ServiceListener oder mit einem ServiceTracker reagiert werden, welcher die entsprechenden Folgeoperationen implementiert.

4.4.1 Programmierung

Da die Grundlage unserer Robotersoftware auf Java basiert, wurden auch viele weitere Teile der Software in Java geschrieben. Zusätzlich wurde auch C und C++ von uns eingesetzt, da diese Sprachen in manchen Bereichen Vorteile gegenüber Java bieten. Als Beispiel ist die Implementierung des Gesichterkennungsalgorithmus zu nennen, der in C++ geschrieben wurde. Die überzeugende Geschwindigkeit der Gesichtserkennung war ein schlagkräftiges Argument, an dieser Stelle C++ einzusetzen. Die Entscheidung, mehrere Programmiersprachen zu nutzen, verlangt nach einer Möglichkeit zwischen den verschiedenen Programmen zu kommunizieren. Hier boten sich uns zwei praxiserprobte Ansätze: Die Kommunikation über

TCP Sockets und

JNI (Java Native Interface)

Beide Varianten wurden implementiert und getestet, um uns letztendlich für eine der beiden Varianten zu entscheiden. Im Folgenden werden die beiden Ansätze beschrieben.

4.4.2 Kommunikation über Sockets

Für die Kommunikation über Sockets müssen beide Kommunikationspartner, hier das Java- und das C-Programm, jeweils ein Socket implementieren, wobei einer die Rolle des Servers und einer die Rolle des Clients einnimmt. Als Test haben wir den Roboter in einem Modus herumfahren lassen, in dem er wahllos die Richtung ändert. Das dafür zuständinge Programm *randomwalk* ist in C++ geschrieben. Über OSGi wird dieses Programm gestartet. Sobald der Roboter mithilfe seiner Laser und dieses Programms ein Hindernis entdeckt, wird über eine bestehende Socketverbindung eine Nachricht an das Javaprogramm geschickt, die dann im OSGi-Framework weiterverarbeitet werden kann. Diese Methode, zwei Programmteile kommunizieren zu lassen, ist einfach zu realisieren, aber ist für unseren Einsatzzweck nicht geeignet, denn aufgrund unserer Softwarestruktur hätten wir sehr viele TCP-Verbindungen offen, die verwaltet werden müssen. Deswegen widmeten wir uns einem weiteren und sehr vielversprechendem Ansatz, dem Java Native Interface.

4.4.3 JNI

Das JNI (Java Native Interface) ist eine Schnittstelle zwischen Java und Funktionen und Programmen, die in anderen Programmiersprachen entwickelt sind. Es bietet dem Java-Programmierer auch die Möglichkeit, auf Features des Betriebssystems zuzugreifen, welche in Java nicht angeboten werden. Um nun Funktionen aus C-Programmen über Java zu nutzen, müssen in der Java-Datei die Funktionen als *native* deklariert werden. Im C-Programm werden diese nativen Methoden implementiert. Dabei muss auf bestimmte Namenskonventionen geachtet werden. Der Programmcode der C-Programme muss in eine dynamisch ladbare Bibliothek gebunden werden. Dazu übersetzen wir das erzeugte C-Programm mit dem gcc Compiler und dem Parameter *-shared*, wodurch wir die Bibliothek, die unter Linux die Dateiendung .so (für *shared object*) besitzt, erhalten. Um die Funktionalität der Bibliothek nutzen zu können, muss diese im Javaquelltext durch den Befehl *System.loadLibrary(lib);* eingebunden werden. Danach kann unter Java auf die in C geschriebenen Methoden zugegriffen werden.
Der Vorteil ist, dass die Kommunikation nicht mehr umständlich über Sockets realisiert wird, sondern der

Methodenaufruf direkt im Javaquelltext stattfindet. Zwar muss bei jeder Änderung im C-Quellcode die *shared library* erneut kompiliert werden, das hat aber wiederum den Vorteil, dass die Bibliothek auch von anderen Klassen genutzt werden kann. Der einfache Aufruf von *System.loadLibrary(lib);* genügt, um die Funktionalität des C-Programms in das Javaprogramm zu integrieren.

4.4.4 Dienste

StatusBundle

Dieses Bundle wurde entwickelt, um Informationen über den Roboter zu ermitteln und globale Konstanten für das System zu verwalten. Andere OSGi-Bundles können die im StatusBundle definierte Schnittstelle *RobotService* benutzen, um auf diese Informationen und Konstanten zuzugreifen.
RobotService ermittelt dabei folgende Informationen über den Roboter:

Roboterkoordinaten Dabei werden vom Player-Server (siehe Anhang D.3) die X- und Y-Koordinaten sowie der Drehwinkel abgefragt.

Bereitschaft der Sensoren Es kann abgefragt werden, ob der Laser und die Kamera funktionsbereit sind, indem vom Player-Server die Bereitschaft des Lasers bzw. vom Kameraserver (siehe Anhang D.1.1) die Bereitschaft der Kamera abgefragt werden.

Aktivität des Bumpers gibt an, ob der Sicherheitsbumper betätigt wurde.

Konfigurationseinstellungen Dazu gehören Port- und Host-Angaben zum Player-Server sowie Informationen über den Kamera-Server.

Weiterhin werden im StatusBundle, wie oben erwähnt, globale Systemkonstanten (*Properties*) abgespeichert. Dadurch wird es möglich, globale Systemeinstellungen an einer zentralen Stelle zu definieren und zu verändern. Systemeinstellungen können zur Laufzeit von anderen Services abgefragt werden.

FaceDetectorService

Der FaceDetectorService sucht Gesichter im aktuellen Kamerabild, Es wird das im Abschnitt 2.3 beschriebene Verfahren verwendet.

findePersonen liefert eine Liste mit Personenhypothesen.

PersonDetectorService

Der PersonDetectorService meldet zwei Klassen mit dem Detector-Interface an. Der erste sucht Ganzkörper, der zweite Oberkörper mit den beschriebenen HOG-Detektoren (Abschnitt 2.4) im aktuellen Kamerabild. Beide geben ein Array mit Detektionen zurück.

findePersonen liefert eine Liste von Personenhypothesen mit deren Dreh- und Neigungswinkel relativ zur Kamera.

LegDetectorService

Der LegDetectorService liefert die durch den Laser erkannten Personen als Array mit Detektionen. In Kapitel 2.2 wird die Funktionsweise des Detektors näher erläutert.

findePersonen liefert eine Liste von Personenhypothesen mit deren Drehungswinkel und Konfidenz.

CamScanBundle

Das CamScan Bundle realisiert die Kamerasteuerung für die Personenedetektion. Die Kamera sieht sich im vorderen Sichtfeld um. Sofern der Beinpaardetektor Personen vermutet, wird zufällig eine ausgewählt und die Kamera dorthin in Drehung und Neigung ausgerichtet.

siehGeradeaus schaltet das Umsehen ein oder aus.

ClusterBundle

Das ClusterBundle realisiert das in Kapitel 2.5 beschrieben Verfahren zum Zusammenfassen von Detektionen. Dazu sammelt das Bundle aktuelle Detektion der vorhandenen Detektoren und liefert eine Liste von Clusterobjekten. Die Clusterobjekte kapseln die zuvor ermittelten Detektionen in Gruppen und liefern zusätzlich Methoden zur Berechnung eines zusammenfassenden Konfidenzwertes und der Position.

findeCluster liefert eine Liste mit zusammengefassten Personenhypothesen aus den einzelnen Detektionen der vier verwendeten Detektoren.

StrategyBundle

In diesem Bundle wurde eine Reihe von Strategien für die Auswahl einer Person bzw. einer von Detektoren gelieferten Hypothese implementiert. Diese Auswahl dient an erster Stelle dazu, im Zustand „Idle“ (siehe Zustandsautomat im Abschnitt 4.2.3) zu entscheiden, ob eine Person gefunden wurde.
Die von Detektoren gelieferte Hypothesen werden im ClusterBundle geeignet zusammengefasst, so dass ein Cluster ein gefundenes Objekt bzw. eine gefundene Person repräsentiert. StrategyBundle bekommt so eine Liste von Clustern, aus der einen passenden Cluster ausgewählt werden soll.
Es wurden folgende Strategien implementiert (eine Person entspricht technisch einem Cluster):

DistanceStrategy wählt eine Person aus, die sich am nächsten zum Roboter befindet.

ProbabilityStrategy wählt eine Person mit der größten Wahrscheinlichkeit aus der entsprechenden Hypothese aus.

RandomStrategy wählt zufällig eine Person aus.

PersonGroupsStrategy identifiziert Gruppen von Personen und findet Personen, die zu keiner Gruppe gehören. Es werden somit die alleinstehenden Personen gefunden. Als Kriterium für die Identifikation solcher Personen wurde der Abstand zwischen den Personen festgelegt. Dabei befinden sich die alleinstehenden Personen am weitesten von allen anderen.

VerificationStrategy verifiziert, dass eine Person in bestimmter Position gefunden werden kann. Als Kriterium für diese Strategie wurde der Abstand einer Person von der übergebenen Position festgelegt. Diese Strategie wird insbesondere benutzt, um zu überprüfen, ob eine Zielperson im Zustand „Zu Person Fahren“ (siehe Zustandsautomat im Abschnitt 4.2.3) weiterhin gefunden werden kann.

Um diese Strategien geeignet zu kombinieren, wurde ***GlobalStrategy*** implementiert, die über eine Schnittstelle ***StrategyService*** von anderen Bundles angesprochen werden kann.

Dynmap

Dieses Bundle ist für den Dynamicmapping-Treiber geschrieben worden und bestimmt, ob und wann ein neuer Weg geplant werden soll. Es ist somit die künstliche Intelligenz hinter dem Algorithmus. Hierbei wird die dynamische Karte benutzt, die der Dynamicmapping-Treiber erstellt. Wenn ein dynamisches Hindernis auftaucht, das der Roboter nicht einfach umfahren kann (wie z.B. eine verschlossene Tür), versucht das Bundle einen neuen Weg zu dem Zielort zu finden, indem der Wavefront-Algorithmus auf der dynamischen Karte mit dem neu eingezeichneten Hindernis aufgerufen wird.
Im Folgenden werden die einzelnen Services des OSGi-Bundles beschrieben:

fahreStrecke lässt den Roboter die angegebene Strecke in die Richtung des angegebenen Winkels fahren.

fahreRoute lässt den Roboter eine Liste von Wegpunkten abfahren. Diese werden durch eine Textdatei eingelesen und abgearbeitet.

fahreKoords lässt den Roboter zu den angegebenen X/Y-Koordinaten mit angegebenen Winkel fahren.

status gibt den Status des Roboters zurück (z.B. „beschäftigt“ oder „idle“).

zielErreicht gibt an ob das Ziel bis auf wenige cm erreicht wurde.

abbruch stoppt den Roboter und bricht die Navigation ab.

CameraService

Der CameraService bewegt die Kamera und liefert Einzelbilder.

bewegeKamera dreht die Kamera an eine bestimmte Position.

holeBild gibt ein Halb- oder Vollbild zurück. Dies wird etwa von der Foto-Funktion benutzt.

macheGeste führt eine der Gesten „Nicken “, „Kopfschütteln “oder „aufgeregt sein “aus.

5 Evaluierung

Im Laufe der Projektgruppe wurden beständig formelle und informelle Tests gemacht, um die Funktion des PARTYBOT zu Testen und Optimieren. Die Navigation und Hindernisvermeidung wurde zunächst mit dem Simulator Stage getestet. Im Anschluss wurden etliche Praxistest gemacht. Die Parameter des Beindetektors wurden mittels einiger Praxistests eingestellt und validiert. Um eine gute und hinreichend schnell berechenbare Konfiguration der HOG-Detektoren zu finden, wurden wiederholt Testaufnahmen mit dem PARTYBOT erzeugt und annotiert. Diese bildeten die Basis für eine ausführliche Reihe von Validierungen. Der Gesichtsdetektor wurde mittels denselben Aufnahmen getestet. Im Anschluss wurden vergleichende Tests mit allen drei bildbasierten Detektoren gemacht, um deren Eignung in verschiedenen Szenarios zu vergleichen. Mit praktischen Versuchen wurde die Entfernungsschätzung der vier Detektoren verglichen. Der Clusteringalgorithmus wurde darauf aufbauend in der Simulation und weiteren Praxistests optimiert. Die auf den Detektionen und Clustern basierenden Strategien wurden, ebenso wie der Zustandsautomat, in Softwaretests überprüft. Die dynamische Hindernisvermeidung und Wegplanung wurde zunächst mit dem Simulator Stage getestet und dann Praxistests unterzogen. In einigen informellen Intergrationstests wurde das Zusammenspiel aller Komponenten optimiert. Schließlich wurde die Gesamtfunktion in einigen Praxistests mit definierten Partyszenarios verifiziert.

5.1 Informelle Tests

Der Schülertag Informatik wurde am 12.11.2009 vom Fachbereich Informatik an der Technischen Universität Dortmund ausgerichtet. Im Rahmen dieser Veranstaltung wurde der Roboter von der Projektgruppe vorgestellt und konnte von den Besuchern bedient werden. Zur Verfügung stand zu diesem Zeitpunkt ein Fotoservice mit der Möglichkeit, das mit der Roboterkamera geschossene Portraitfoto an eine Mailadresse gesendet werden. Die Besonderheit war hier, dass sich die Kamera automatisch auf Gesichter fokussieren konnte. Des Weiteren konnte der Roboter frei fahren und dabei Hindernissen ausweichen und auf Wunsch Personen verfolgen. Außerdem war es möglich auf einer Karte eine Route durch Wegpunkte vorzugeben, welche dann vom Roboter abgefahren wurden. Um all diese Funktionen zu steuern, wurde speziell für den Schülertag eine grafische Nutzeroberfläche entwickelt, welche per Touchscreen des Roboters bedient werden konnte. Die gesamte Funktionalität war bereits mit Hilfe des OSGi-Frameworks realisiert. Das System lief trotz der großen Ansammlung von Personen sehr stabil. Abbildung 32 zeigt ein Foto vom Schülertag.

5.2 Konfiguration und Evaluierung der HOG-Basierten Detektoren

Mustererkennung ist eine komplexe Aufgabe, bei der die ersten Schritte die wichtigsten sind. Was man am Anfang bei der Gewinnung der Merkmale falsch macht, kann man im Folgenden kaum wieder gut machen. Jeder Schritt ist von allen vorhergehenden abhängig und man weiß erst am Ende, also beim Ermitteln der durchschnittlichen Erkennungsrate, wie gut das Ganze wirklich gelungen ist. Am Rechner hat man die Möglichkeit, die Ergebnisse vorhergehender Schritte wiederzuverwenden, damit ergeben sich mögliche Desingzyklen wie in Abbildung 33 skizziert [14, S. 14ff]. Zur Optimierung des Gesamtergebnisses bei Variationen ist streng genommen stets eine Veränderung der Parameter in allen späteren Stufen erforderlich. Um Zahle der Neubeginne des Designzyklus zu minimieren, ist große Sorgfalt bei der Implementierung

Abbildung 32: Der PARTYBOT auf dem Schülertag

und Konfiguration, vor allem der frühen Schritte, geboten. Optimierungen der Rechenzeit sind somit erst spät sinnvoll, allerdings helfen sie während des Designprozesses, die durchaus sehr langwierigen Evaluierungsläufe[3] zu beschleunigen und damit die Evaluierung von mehr Parametrisierungen in vertretbarer Zeit möglich zu machen.

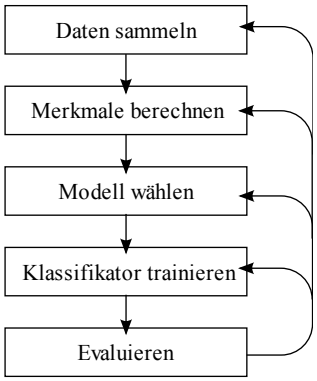


Abbildung 33: Entwurfszyklen eines Mustererkennungsystems

Zum Finden optimaler Modelle und Suchparameter wurde im Laufe der PG dementsprechend eine große Zahl Validierungen durchgeführt. Dabei wurden zunächst die Parameter der HOGs und Klassifikatoren anhand von Kreuzvalidierungen mit annotierten Bildbereichen überprüft. Anschließend wurde dann erschöpfende Suche über Kamerabildserien angewendet, um Suchstrategien und Nachverarbeitungsschritte zu optimieren. Für Rechenzeitvergleiche wurde ein fixer Standard-PC[4] verwendet, der in der Leistung mit der Zielplattform auf dem SCITOS vergleichbar ist. Grobe Zielvorgabe war eine Rechenzeit von einer Sekunde

[3] in unserem Fall bis zu 4 Stunden für einen einzigen Parametersatz
[4] AMD Athlon X2 3800+, 3GB RAM, Ubuntu Linux 8.10

pro Kamerabild bei einer vertretbaren Anzahl von Fehldetektionen (englisch *false positives*, kurz FP). Es wurden die folgenden Parameter aufeinanderfolgend optimiert:

HOG-Konfiguration Die Anzahl von Zellen und gemeinsam normierten Blöcken.

Klassifikator Typ und Parameter des verwendeten Musterklassifikators.

Suchraum Schrittweite, in der die Bilder abgesucht werden und heuristische Einschränkung der betrachteten Positionen.

Clustering Parameter des Mean-Shift-Clusterers (Abschnitt 2.4.2), der mehrere sich überlappende Detektionen zusammenfasst.

5.2.1 Vorgehensweise

Für die Validierung wurden mit dem Recorder-Tool (Anhang S. 118) Bilderserien mit der Kamera inklusive ihrer Positionsdaten aufgenommen. Diese wurden in einer gemeinsamen Anstrengung von allen PG-Teilnehmern annotiert. Dabei wurden rechteckige Annotationen mit den Namen der aufgenommen Menschen für vollständig sichtbare Ganz- und Oberkörper versehen. Ambivalente Fälle wie halbe Personen, Spiegelungen etc. wurden mit einem Fragezeichen annotiert. Um Restklassenbeispiele für die Kreuzvalidierung zu erhalten, wurden zufällig Rechtecke gewählt, welche keine Annotation mehr als 30% überschneiden.
Zur Klassifikation wurden sowohl Multilayer-Perzeptrons (kurz MLPs, siehe [6]), wie auch die *Stützvektormethode* (englisch *support vector machine*, kurz SVM, siehe [9]) verwendet. Während bei der SVM die übliche Kreuzvalidierung durchgeführt wurde, war bei den MLPs eine spezielle Anpassung nötig: Da der verwendete Trainingsalgorithmus nur brauchbare Ergebnisse liefert, wenn während des Training bereits eine Validierung durchgeführt wird, war die weitere Unterteilung der Trainingsmenge nötig. Wir wählten dazu pro Durchlauf ein weiteres festes Validierungsset, wie in Abbildung 35 dargestellt. Darüber hinaus wurde wegen der stark schwankenden Ergebnisse die Validierung für die MLPs fünfmal auf jedem Datensatz wiederholt. Unglücklicherweise bietet die eingesetzte Bibliothek keine Möglichkeit, die Randomisierung der Eingabedaten zu fixieren oder zu unterbinden. Darüber hinaus schwankt die Qualität des MLP-Trainings sehr stark, insbesondere bei schlechten Parametern.
Die SVMs werden ebenfalls mit einem passenden Gradientenabstiegsverfahren trainiert. Bei der SMO (englisch *sequential minimal optimization* siehe [47]) werden sukzessive Paare der einzustellenden α_i optimiert. Dieses Verfahren konvergiert immer zu einem Optimum für die Trainingsstichprobe, jedoch variiert die konkrete Wertebelegung und damit die Eignung für unbekannte Daten auch mit der Randomisierung der Eingabedaten. Die *libsvm* überlässt diesen Schritt jedoch dem Anwender und bietet damit zusätzlich den Vorteil exakt reproduzierbarer Ergebnisse.
Insbesondere ermöglicht der Verzicht auf Anwendung einer Kernelfunktion die Beurteilung der linearen Trennschärfe der Merkmale [13]. Bei der erschöpfenden Suche wurden entsprechende Annotationen bewertet. Eine mindestens 50%ige Überdeckung mit einer Annotation gilt als Treffer, sofern sie nicht kleiner als 50% der annotierten Fläche ist. Detektionen mit über 30% Überlappung mit einer Fragezeichenregion wurden von der Wertung ausgenommen (Abbildung 36).

Abbildung 34: Kreuzvalidierung

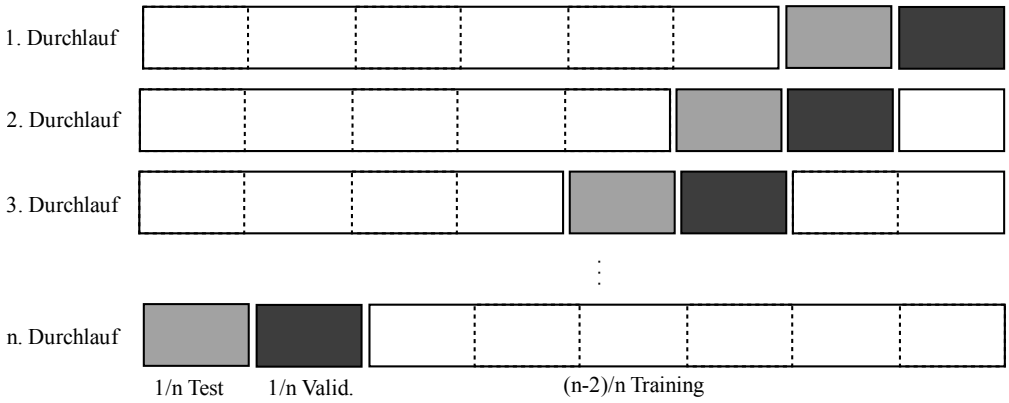


Abbildung 35: Kreuzvalidierung mit Subvalidierung für MLP

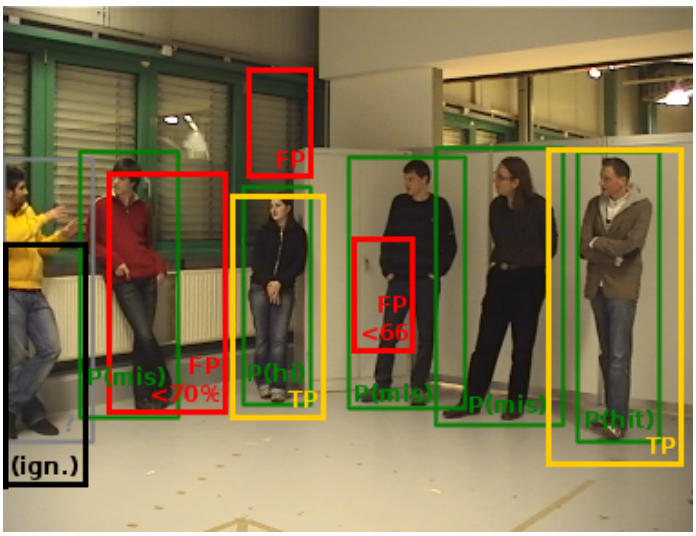


Abbildung 36: Beispiel zur Bewertung von Detektionen abhängig von Annotationen: 2/5 Trefferrate, 3 Fehldetektionen (FP), 2 Richtige Detektionen (TP), 1 Ignorierte Detektion (ign.)

Zur Bewertung der Ergebnisse mit erschöpfender Suche, insbesondere zum Vergleich von Clusteringstrategien, sind die auf Detektionsanzahlen basierenden Maße wenig geeignet. Die Menge der möglichen geclusterten Detektionen ist nur schwer zu bestimmen. Ohne reale Nichtdetektionen ist die Berechnung von Fehler, F1 etc. wenig nützlich. Die Genauigkeit (englisch *precision*) hingegen steigt sowohl mit Mehrfachdetektionen wie auch bei Nichtdetektionen. Beides entspricht unerwünschtem Verhalten.

$$\text{Genauigkeit} := \frac{TP}{TP + FP} \tag{23}$$

$$\text{Vollständigkeit} := \frac{TP}{TP + FN} \tag{24}$$

Der F1 Wert ist das harmonische Mittel der Vollständigkeit und der Genauigkeit. Er wird als Maßzahl für den besten Kompromiss verwendet.

$$\text{F1} := \frac{2 \cdot \text{Vollständigkeit} \cdot \text{Genauigkeit}}{\text{Vollständigkeit} + \text{Genauigkeit}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{25}$$

Um sinnvolle Aussagen über die Qualität des Clusterings machen zu können, wurden die Hilfsgrößen *Trefferrate* und *Nichtredundanz* berechnet.

$$\text{Trefferrate} := \frac{\#\text{AbgedeckteAnnotationen}}{\#\text{Annotationen}} \tag{26}$$

$$\text{Nichtredundanz} := \frac{\#\text{AbgedeckteAnnotationen}}{\#\text{TreffervonAnnotationen}} \tag{27}$$

Dabei spiegelt die Trefferrate die Anzahl der getroffenen gewünschten Detektionen wider, d.h. sie erreicht 100% genau dann, wenn alle gewünschten Ganz- / Oberkörper von einer Detektion abgedeckt werden. Sie entspricht der Vollständigkeit (englisch *recall*) gemessen an Personen pro Bild. Zur bessern Unterscheidbarkeit wird im Folgenden der Begriff Vollständigkeit für Detektionen und Trefferrate für Personen verwendet. Die Nichtredundanz ist der Kehrwert der Mehrfachabdeckung, d.h. sie erreicht 100% genau dann, wenn jede gefundene Person von genau einer Detektion getroffen wird.

5.2.2 HOG Konfigurationen

Zur Bestimmung der optimalen HOG-Parameter wurden verschiedene Kreuzvalidierungen und Tests mit erschöpfender Suche durchgeführt. Die Funktionsweise der orientierten Gradientenhistogramme wurde in Abschnitt 2.4.2 (S. 26ff.) erläutert. Da die Normalisierungsvorschrift bei Personen quasi keinen Einfluss auf die Qualität der Merkmale hat [12, S. 6], wurde schlicht die L2-Norm verwendet, welche sich für viele natürliche Objekte bewährt hat [13, S. 47ff.]. Die Anzahl von neun Richtungen für die Gradienten wurde ebenso übernommen.
Für die Aufteilung in überlappende Blöcke aus Zellen wurden für Ganz- und Oberkörper verschiedene Varianten getestet. Auch hier zeigten sich wenig deutliche Unterschiede. Lediglich der Verzicht auf mehrere, getrennt normierte Blöcke ist deutlich schlechter. Da sich in der Kreuzvalidierung keine signifikanten Unterschiede ergaben, wurden die Aufteilungsschemata auch mit linearen SVMs per erschöpfender Suche verglichen. Abbildung 37 zeigt die Ergebnisse für Ganzkörper. Hier ist die Variante mit 3x7 Blocks aus 3x3 Zellen mit je einer Zelle Überlappung verwandten Aufteilungen leicht überlegen. Dies bedeutet, dass auf

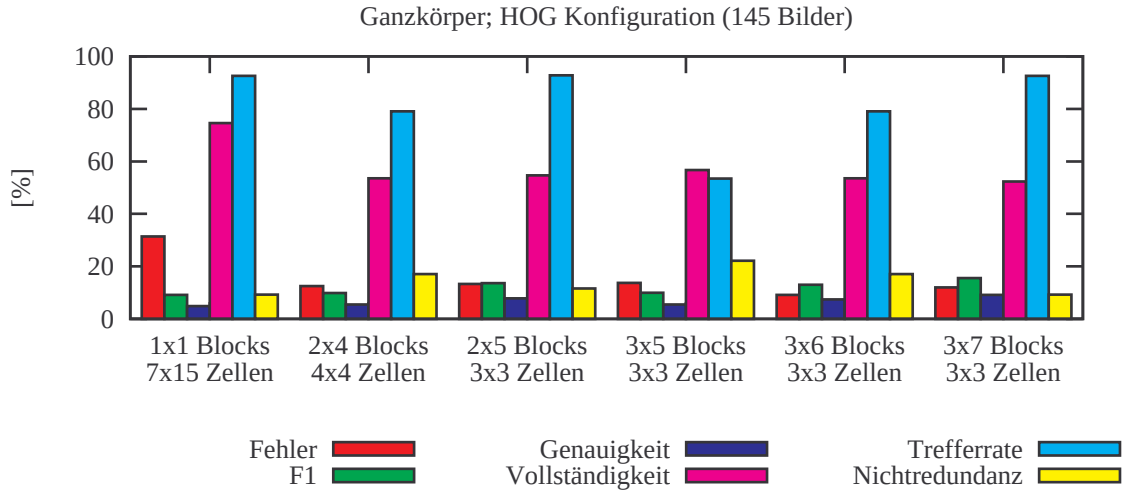


Abbildung 37: Evaluierung von HOG Konfigurationen für Ganzkörper

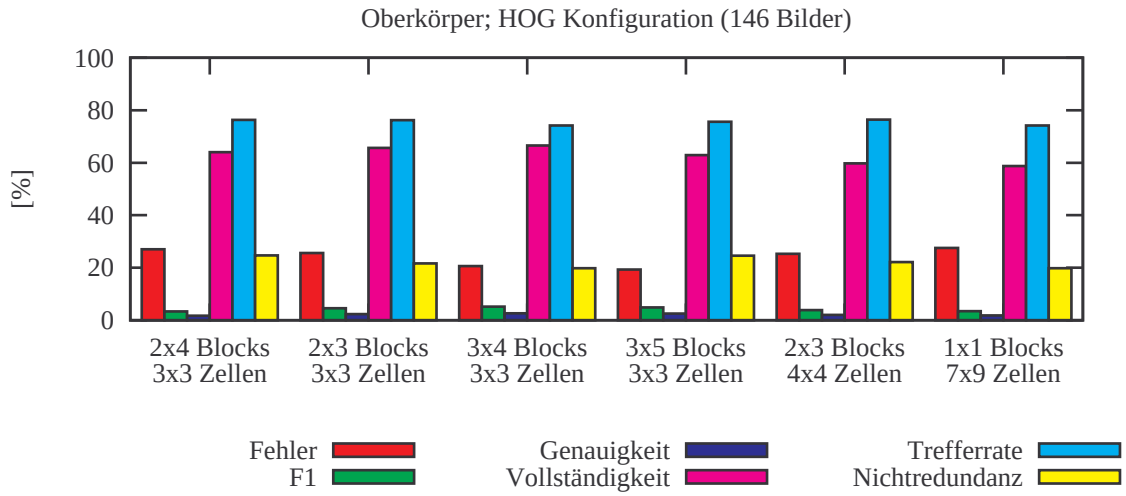


Abbildung 38: Evaluierung von HOG Konfigurationen für Oberkörper

einem Suchfenster aus $3 \cdot (3-1)+1$ x $7 \cdot (3-1)+1$ = 7 x 15 Zellen $3 \cdot 3 \cdot 3 \cdot 7 \cdot 9$ = 1.701 Histogrammwerte berechnet werden. Abbildung 38 zeigt die Ergebnisse für Oberkörper. Hier wurden 3x5 Blöcke aus 3x3 Zellen als Konfiguration gewählt. Dies entspricht $3 \cdot (3-1)+1$ x $5 \cdot (3-1)+1$ = 7 x 11 Zellen und $3 \cdot 3 \cdot 3 \cdot 5 \cdot 9$ = 1.215 Histogrammwerten.

5.2.3 Klassifikatoren

Wir haben sowohl SVMs als auch MLPs als Klassifikator eingesetzt. Die verwendeten Implementierungen zeigten verschiedene Stärken und Schwächen. Bei der Evaluierung waren hohe Stabilität und Aussagekraft sowie kurze Trainingszeiten gefordert. Für den Einsatz im Zielsystem waren in erster Linie gute Ergebnisse bei kurzer Klassifizierungszeit wichtig.

SVM

Ein derzeit oft verwendetes Werkzeug zur Klassifikation ist die *Stützvektormethode*[5]. Hierbei handelt es sich zunächst um eine lineare Separation von Daten in zwei Klassen. Das berechnete Modell beschreibt eine separierende Hyperfläche H aus der Linearkombination von einem Teil der Eingabedaten, den sogenannten Stützvektoren. Zusätzlich kommt statt des Skalarproduktes oft eine Kernelfunktion $k(\cdot)$ zum Einsatz, eine Abbildung aus dem Merkmalsraum – etwa Polynomen n-ten Grades, Gauß- oder Sigmoidfunktionen – in einen höherdimensionalen Raum. Die lineare Klassifikation wird im Bildraum vorgenommen. Die Klassifikations-Hyperfläche h wird dann in den Merkmalsraum zurücktransformiert. Diese $k^{-1}(h)$ ist dort dann im Allgemeinen nichtlinear.

$$H: \quad k(\vec{x},\vec{w})+b \quad = \quad 0 \quad = \quad \sum_{i=1}^{N} k(\vec{x},\vec{x}_i)\alpha_i y_i + b \tag{28}$$

Wegen ihres deterministischen und schnellen Trainings wurden Typ1 SVMs ohne Kernelfunktion zur Evaluierung eingesetzt. Wie Tabelle 3 zeigt, führen Kernelfunktionen schnell zu vielen Supportvektoren und sehr viel langsamerer Klassifikation. Typ2 SVMs waren grundsätzlich langsam ohne Qualitätsgewinn. Die Sigmoid-Kernel in Typ1 SVMs zeigen als einzige geringere Fehlerraten als die lineare SVM. Diese sind neuronalen Netzen nachempfunden, jedoch mit 23ms per Auswertung zehnmal langsamer als ein vergleichbares MLP. Die schnellste Variante der linearen Typ1 SVM benötigt ca. die dreifache Rechenzeit, was sie in Anbetracht der zeitkritischen Anwendung für den Einsatz auf dem PARTYBOT disqualifiziert.

MLP

Die alternativ eingesetzten MLPs wurden in Abschnitt 2.4.2 (S. 28) erläutert. Wie die lineare SVM leistet selbst ein Netz ohne Zwischenschichten mit nur zwei Ausgabeneuronen eine akzeptable Klassifikationsleistung. Diese lässt sich durch die Hinzunahme von Zwischenschichten durchaus noch steigern. Beim direkten Vergleich waren Netze mit zwei inneren Schichten stets überlegen. Abbildung 39 zeigt das Ergebnis einer Kreuzvalidierung über Oberkörper von acht Personen. Das heißt, dass in der Menge der Bildserien acht

[5] eine verständliche Einführung findet sich etwa in [9]

Typ 1, Kernel	linear	poly. (Grad 3)	radial	sigmoid
F1 [%]	88,27±12,0	88,23±11,8	88,12±12,0	90,88± 8,4
Akkuranz [%]	93,66± 6,2	93,54± 6,1	93,54± 6,3	94,70± 4,6
Genauigkeit [%]	100,00± 0,0	98,29± 0,3	98,73± 0,3	97,81± 0,9
Vollständigkeit [%]	80,99±18,5	82,02±18,5	81,65±19,0	86,07±14,4
Fehler [%]	6,34± 6,2	6,46± 6,1	6,46± 6,3	5,30± 4,6
SV ($\alpha_i \neq 0$)	370,80±21,0	889,40±66,7	840,20±51,0	1.643,00±61,2
Zeit [ms]	6,59± 0,3	12,91± 1,8	12,68± 1,0	22,90± 2,0

Typ 2, Kernel	linear	poly. (Grad 3)	radial	sigmoid
F1 [%]	90,06± 7,6	88,51±10,5	89,33± 8,8	91,26± 5,9
Akkuranz [%]	93,82± 4,2	93,05± 5,4	93,43± 4,8	94,62± 3,3
Genauigkeit [%]	92,45± 2,4	90,86± 2,8	91,87± 3,2	95,43± 1,2
Vollständigkeit [%]	88,56±12,3	87,67±16,2	87,81±13,7	88,14±10,8
Fehler [%]	6,18± 4,2	6,95± 5,4	6,57± 4,8	5,38± 3,3
SV ($\alpha_i \neq 0$)	2.571,60± 2,2	2.563,60± 4,3	2.574,60± 5,4	2.583,60± 5,2
Zeit [ms]	35,97± 1,9	33,12± 1,9	35,16± 1,9	36,51± 4,9

Tabelle 3: Vergleich von SVM Kernen und Typen (Ganzkörper; 1:4 xy, N=4250)

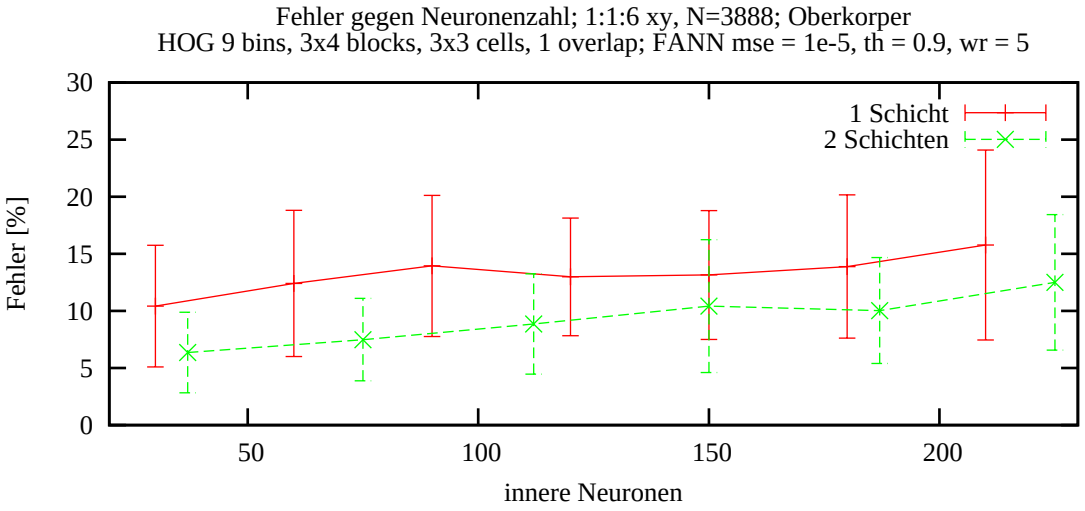


Abbildung 39: Variation der Netztopologie

verschiedene Personen zu sehen sind. Es werden insgesamt 8 Validierungsläufe durchgeführt, wobei immer mit sechs Personen trainiert, mit einer getestet und mit der letzten Person validiert wird. Es zeigte sich, dass schon sehr kleine Netze geringere Fehlerraten aufwiesen. Für die gegebenen Daten ist also ein eher allgemeines Modell von Vorteil, bei vielen Neuronen neigt der Klassifikator zur Überspezialisierung. Entsprechend wurden als Endkonfiguration kleine zweischichtige Netze eingesetzt[6].

[6] 1215 Eingänge gefolgt von Schichten aus 60 und 15 Neuronen für Oberkörper und 1701 Eingänge gefolgt von Schichten aus 84 und 21 Neuronen für Ganzkörper

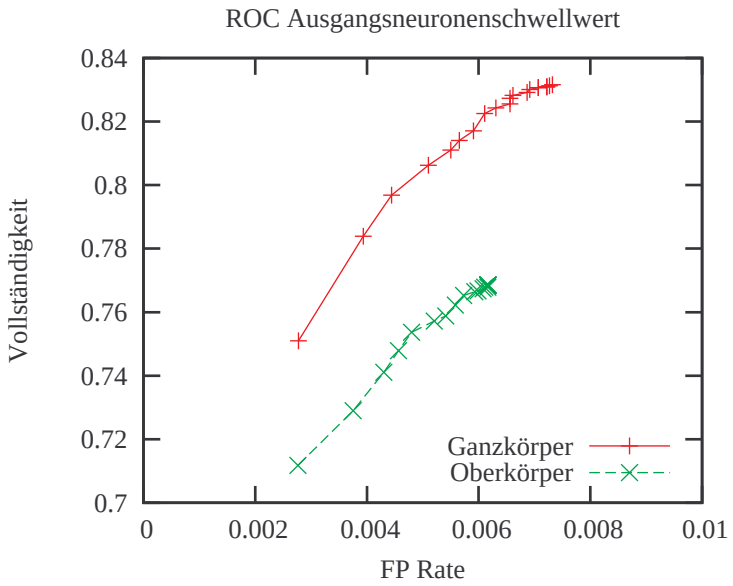


Abbildung 40: Variation des Schwellwertes der Ausgabeneuronen

Entscheidungsfunktion

Die Klassifikation wird nach dem Vergleich der Werte der Ausgabeneuronen o_i mit einem Schwellwert getroffen. Dabei muss das „ist Person“ Neuron o_0 über dem Schwellwert liegen und einen deutlich höheren Wert haben als das „ist keine Person“ Neuron o_1. Die Entscheidung erfolgt nach der in Gleichung 29 angegebenen Formel getroffen.

$$y = \begin{cases} 1 & \text{falls } o_0 > t \wedge o_0/o_1 > 5 \\ 0 & \text{sonst} \end{cases} \tag{29}$$

Abbildung 40 zeigt die Auswirkung der Variation des Schwellwertes t. Die Auswertung erfolgt auf des Basis von über 20.000 Nichtpersonenbilder und über 3.000 Personenbildern, welche zufällig aus den annotierten Testaufnahmen ausgewählt wurden. Bei der erschöpfenden Suche werden über 2.000 Detektionsfenster pro Bild betrachtet, von denen meint nur 20 im Bereich einer Person liegen. Um dem Einfluss der ungleichen *a priori* Wahrscheinlichkeiten nicht unberücksichtigt zu lassen und hinreichend viele Ausprägungen der „nicht Personen“ zu erfassen wurden bei diesen Evaluierungen eine fünf bis zehnfach größere Menge von Negativbeispielen verwendet.

5.2.4 Suchraum

Nach der Wahl der Klassifikatoren und HOG-Konfiguration war das nächste Ziel, den verwendeten Suchraum zu optimieren. Ziel der Optimierung war den Suchraum so klein wie möglich und bei guter Abdeckung und damit guten Klassifikationsergebnissen zu gestalten. Für alle Tests wurde eine heuristische Einschränkung der möglichen Personenpositionen verwendet, da diese den Zeitaufwand signifikant reduziert.

Objekt	Ganzkörper				Oberkörper			
Einschränkung	–		Fußboden		–		Fußboden	
Clustering	–	5,2	–	5,2	–	9,2	–	9,2
Genauigkeit [%]	37,41	57,39	43,82	60,56	23,98	37,07	37,07	47,37
Trefferrate [%]	100,00	52,26	100,00	30,28	100,00	42,72	100,00	39,89
Nichtredundanz [%]	0,65	51,65	0,90	52,84	0,59	46,13	0,85	46,79
Detektionen pro Bild	4.674	3	1.824	2	5.835	4	2.105	3
Zeit pro Bild [s]	2,167	1,953	0,879	0,817	0,780	1,706	0,329	0,565

Tabelle 4: Vergleich der Ergebnisse mit und ohne Fußbodeneinschränkung

Schrittweite

Bei der erschöpfenden Suche und dem finalen Detektor wird die Schrittweite, mit der das Detektorfenster über das Bild geschoben wird, festgelegt. Das Suchfenster wird in horizontaler wie vertikaler Richtung um Zellbreiten verschoben, welche ihrerseits in logarithmisch äquidistanten Stufen, den möglichen Entfernungen entsprechend, skaliert werden. Es ergibt sich ein pyramidaler Bereich vom Faktor $s = 1 \ldots 3.2$ und $57 \ldots 13$ horizontale sowie $33 \ldots 0$ vertikale Zellen von $6 \cdot s$ Pixeln. Für Oberkörper wurden Versatzschritte von 1,5 bis 0,5 Zellen mit 3-11 Tiefenebenen mit erschöpfender Suche verglichen. Die Berechnung wird mit einem mittleren Netz auf dem Referenz-PC in 0,5 bis 5,5fps[7] durchgeführt. Nach den Ergebnissen in Abbildung 42 wurde ein Versatz von einer Zelle ($6 \cdot s^n$ Pixeln) mit neun Tiefenebenen gewählt. Die neun Tiefenebenen entsprechen jeweils dem maximalen F1 Wert (etwa 21,16 zu 20,95 und 20,88 bei 7 und 11 Ebenen). Feinerer Zellenversatz führt zu keiner signifikanten Verbesserung, und die Rechenzeit liegt mit 1,7 fps (0,588 spf) im gewünschten Bereich. Für Ganzkörper ergibt sich ein ähnliches Bild (Abbildung 41). Hier wurden 7 Ebenen gewählt. Diese Ebenenwahl entspricht in beiden Fällen einem Skalierungsfaktor von ca. 1,2. Der von Dalal [13] empfohlene Skalierungsfaktor von 1,05 - 1,01 entspricht 35 - 170 Ebenen, dies ist derzeit nur in akademischen Untersuchungen verwendbar da die Laufzeit sehr hoch ist.

Einschränkung

Die Suchraumeinschränkung auf Personen, die auf dem Fußboden stehen, wurde zur Reduktion der Rechenzeit und Fehldetektionen eingeführt. Als initiale Überprüfung der berechneten Fußbodenposition nach Gleichung 15 wurden Testaufnahmen eines Maßbandes auf den Fußboden in verschiedenen Kamerawinkeln gemacht (Abbildung 43). Wie in Tabelle 4 zu sehen ist, wird die Zahl der betrachteten Suchfenster sowie die Rechenzeit mehr als halbiert. Zur Erfassung des tatsächlichen Rechenzeitgewinns wird auch die Zeit mit dem im Folgenden optimierten Clustering tabelliert.

5.2.5 Clustering

Da prinzipiell überlappende Detektionen auftreten, ist ein Clustering der Ergebnisse unvermeidlich. Wir verwenden den Mean-Shift-Cluster Algorithmus [11] zum Zusammenfassen benachbarter Detektionen. Wesentliche Parameter sind der verwendete Kernel, dessen Bandbreite, die Schwellwerte für Zusammenfassung

[7] Bilder pro Sekunde (englisch *frames per second*, kurz fps)

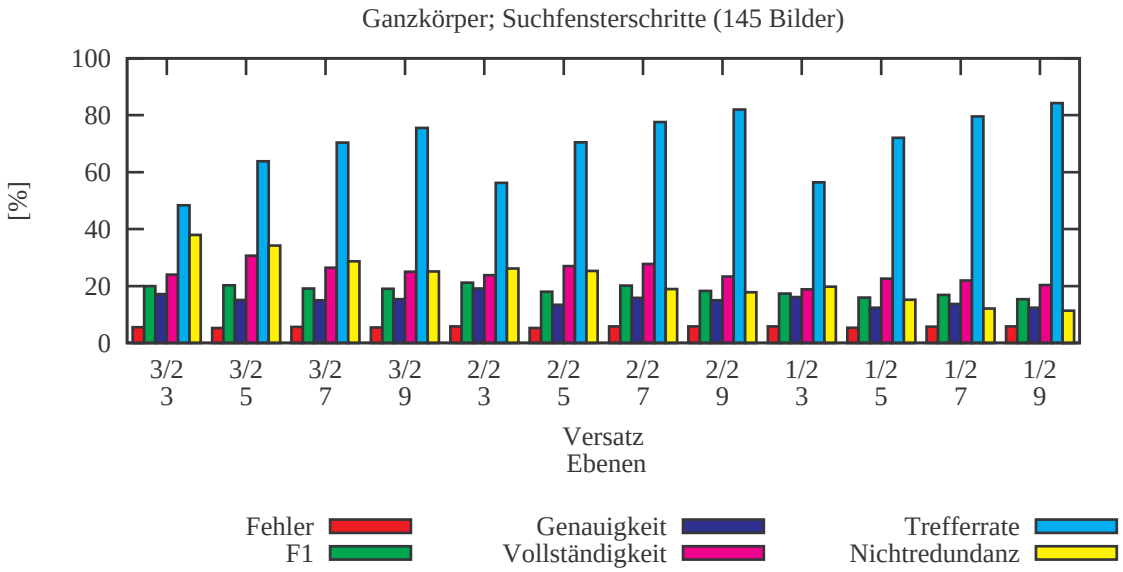


Abbildung 41: Evaluierung von Suchraumschritten für Ganzkörper

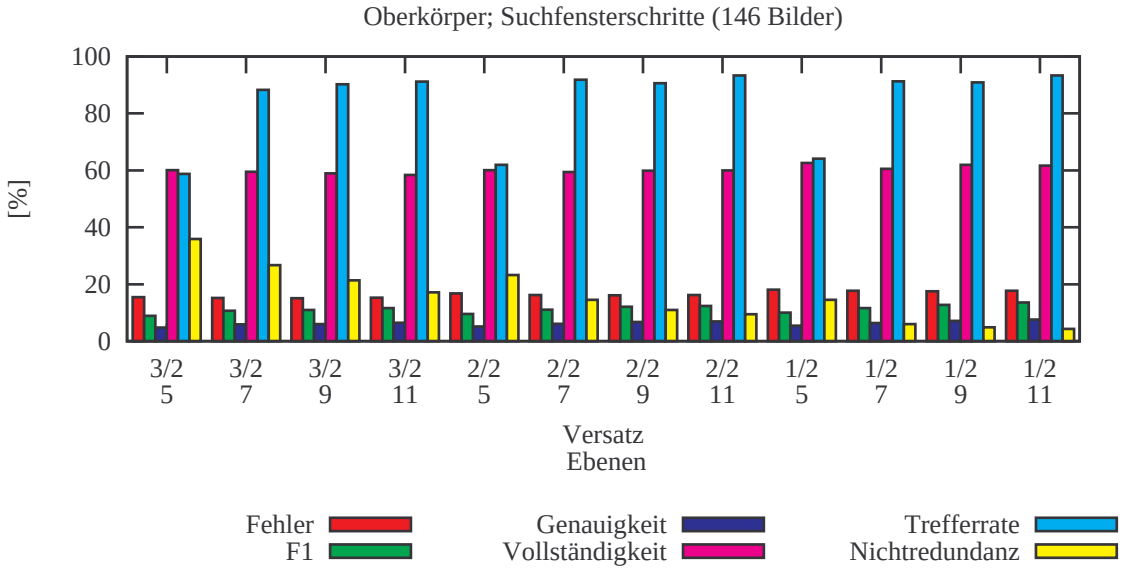


Abbildung 42: Evaluierung von Suchraumschritten für Oberkörper

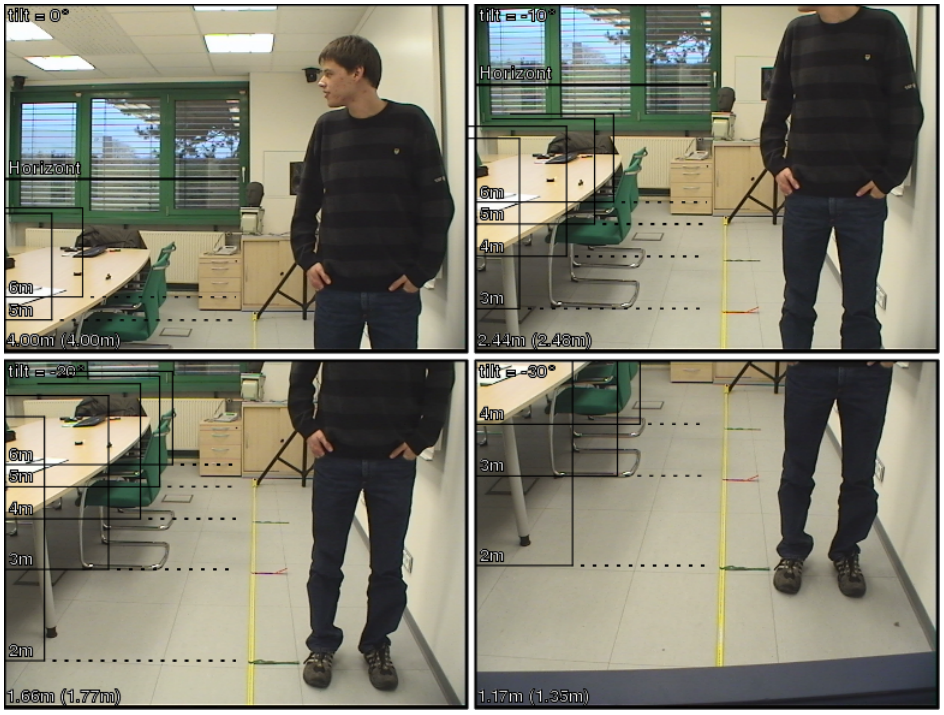


Abbildung 43: Test der Abschätzung der Fußbodenhöhe

und schließlich die minimale Anzahl von Einzelergebnissen (englisch *support*), welche einem Cluster zugrunde liegen müssen.

Funktion

Es wurden vielerlei Evaluierungen über diesen Parameterraum gemacht, wobei in der Regel jeweils ein einzelner Parameter optimiert wurde, um dann von dessen Optimum einen weiteren zu optimieren (eine Art manueller Bergsteigeralgorithmus). Ein Gaußkernel erwies sich als gut geeignet.
Da sich die Laufzeit bei geringerem Abbruchschwellwert nicht signifikant erhöht, der Fehler aber reduziert werden konnte, wurde ein geringer Wert von 0,1 gewählt. Der Verschmelzungsschwellwert wurde auf 0,8 eingestellt, da sowohl geringere als auch höhere Werte die Trefferrate reduzieren.
Es war notwendig, die Bandbreite (Standardabweichung) der Gaußfunktion zu variieren, um das Maß der Zusammenfassung einzustellen. Hierbei wurden alle Ergebnisse ohne besonderen Schwellwert zusammengefasst. Wie in Abbildung 46 zu erkennen ist eine Basisbandbreite von 8-10 Pixeln ein guter Wert für Oberkörper.[8] Hier werden die Fehler abgesenkt, ohne zu viele Treffer zu verlieren. Für Ganzkörper ergibt sich ein Wert von 10-12 Pixeln (Abbildung 44).

Unterstützung

Um Rechenzeit zu sparen und gleichmäßigere Ergebnisse zu erhalten, wurde gegenüber dem ersten Ansatz, stets alle Detektionen als Ganzes zusammenzufassen, ein zweistufiges Verfahren eingeführt. Zunächst wer-

[8] Das Fragezeichen in der Bildüberschrift bezeichnet den im Diagramm variierten Parameter.

Clustering	aus	einstufig			zweistufig		
min, support	–	5	10	15	3,2	5,2	7,2
Genauigkeit der Detektionen pro Bild [%]	43,82	58,63	57,34	54,96	63,91	60,56	38,79
Trefferrate der Personen pro Bild [%]	100,00	62,07	43,18	29,18	56,79	30,28	9,14
Nichtredundanz [%]	0,90	50,69	47,89	49,99	50,42	52,84	63,39
Detektionen pro Bild	1.824	3	2	1	3	2	1
Zeit pro Bild [s]	0,912	0,901	0,913	0,874	0,836	0,811	0,816

Tabelle 5: Evaluierung von Clustering-Schwellwerten für Ganzkörper

Clustering	aus	einstufig			zweistufig		
min, support	–	10	15	20	7,2	9,2	11,2
Genauigkeit der Detektionen pro Bild [%]	37,07	51,28	52,97	52,78	48,47	47,37	43,41
Trefferrate der Personen pro Bild [%]	100,00	41,41	40,33	40,33	42,17	39,89	27,68
Nichtredundanz [%]	0,85	43,47	43,33	44,30	43,99	46,79	45,84
Detektionen pro Bild	2.105	3	3	3	4	3	2
Zeit pro Bild [s]	0,329	2,114	2,111	2,041	0,597	0,565	0,570

Tabelle 6: Evaluierung von Clustering-Schwellwerten für Oberkörper

den die Detektionen einer Skalierungsstufe zusammengefasst. Danach werden die übriggebliebenen Rechtecke erneut zusammengefasst. Wie in Tabelle 5 und 6 zu erkennen ist, erzielt dieses Verfahren in kürzerer Zeit gleich gute bis bessere Ergebnisse, führt aber zu einer weiteren Quantisierung der Einstellmöglichkeiten. Für Oberköperdetektions-Netze durchschnittlicher Qualität reduziert sich der zusätzliche Zeitaufwand von 1,8 auf 0,23s. Da die Detektionen insgesamt wenig Trennschärfe haben, kommt es zu vielen Fehldetektionen, die vom Clustering unterdrückt werden müssen. So ist der Geschwindigkeitsgewinn des zweistufigen Verfahrens wegen der vielen Detektionen sehr deutlich. Die später eingesetzten, mehrfach optimierten Oberköperdetektoren verhalten sich stärker selektiv, so dass der Zeitgewinn hier zurückgeht. Bei den Ganzkörperdetektionen ist der Zeitgewinn hingegen immer marginal.

Nachdem die anderen Parameter fixiert sind, wurden die Schwellwerte für die Unterstützung (englisch *support*) eingestellt. Da die Fehler mit dem Unterstüzungsschwellwert fallen, ist der Punkt optimal, bei dem noch viele Detektionen über der Schwelle liegen. Entsprechend wurde der Wert vor zu starkem Abfall der Trefferrate gewählt, welcher mit einem sehr hohen Genauigkeitswert zusammenfällt.

Bei den Ganzkörperdetektoren schneiden sich die Werte für Trefferrate und Genauigkeit bei ca. 75% im Teil-Szenario „White Room“ mit den in den vorhergehenden Abschnitten ermittelten Werten bei (3, 2) wie Abbildung 45 zu sehen ist. Dasselbe tritt beim einstufigen Clustern bei einem Unterstützugsschwellwert von 8 ein. Ein höherer Schwellwert als 0,95 für die Ausgabeneuronen führt zu weniger Fehldetektionen im Vorfeld, so dass etwa für 0,99 nur noch einstufiges Clustern mit einem Unterstützungsschwellwert von 3 über 70% Genauigkeit und Trefferrate erzielt. Für Oberkörper ist ein guter Wert (3, 1) oder ein einstufiges Clustern mit einer Schwelle von 5 (vgl. Abbildung 47).

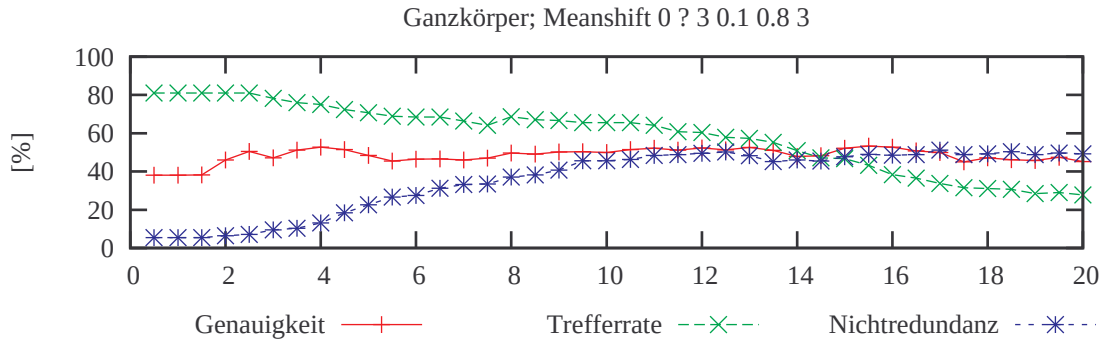
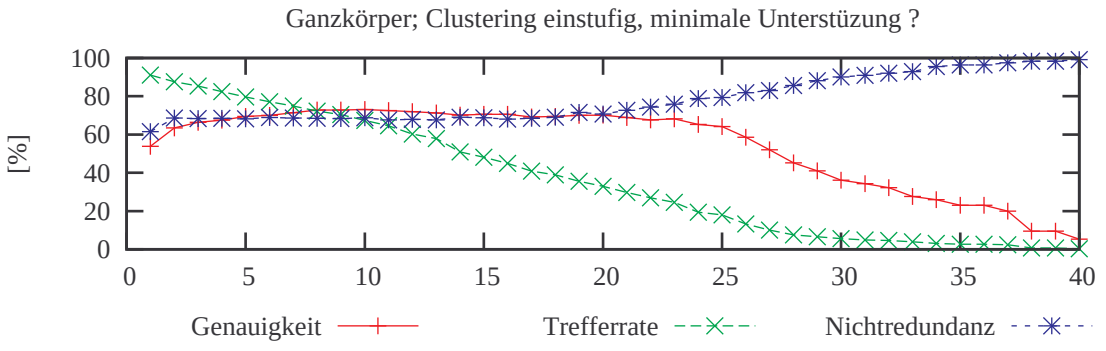
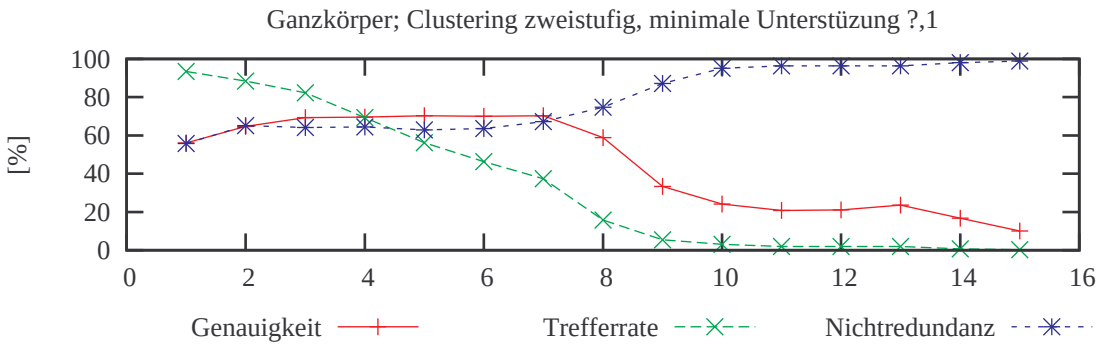


Abbildung 44: Evaluierung von Mean-Shift-Cluster Gaußkernel-Bandbreite (σ) für Ganzkörper





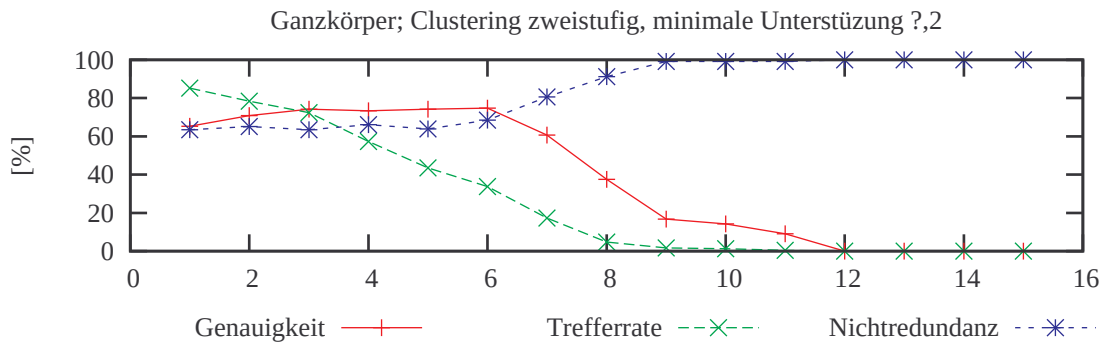


Abbildung 45: Evaluierung von Mean-Shift-Cluster Schwellwerten für Ganzkörper

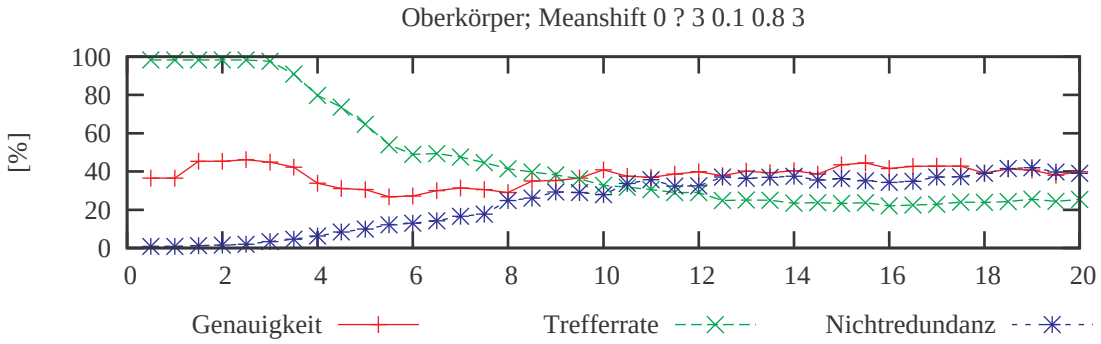
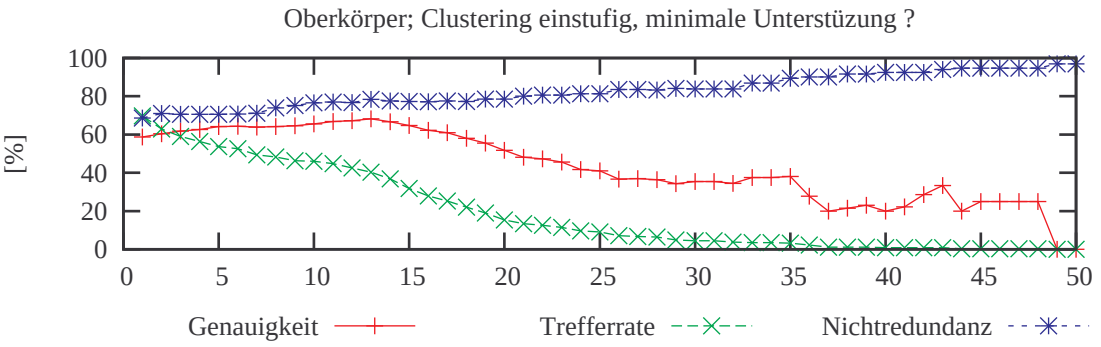
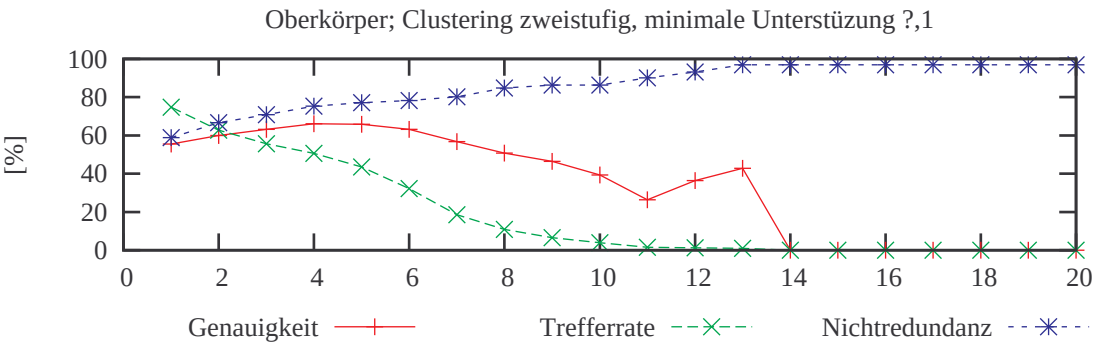


Abbildung 46: Evaluierung von Mean-Shift-Cluster Gaußkernel-Bandbreite (σ) für Oberkörper





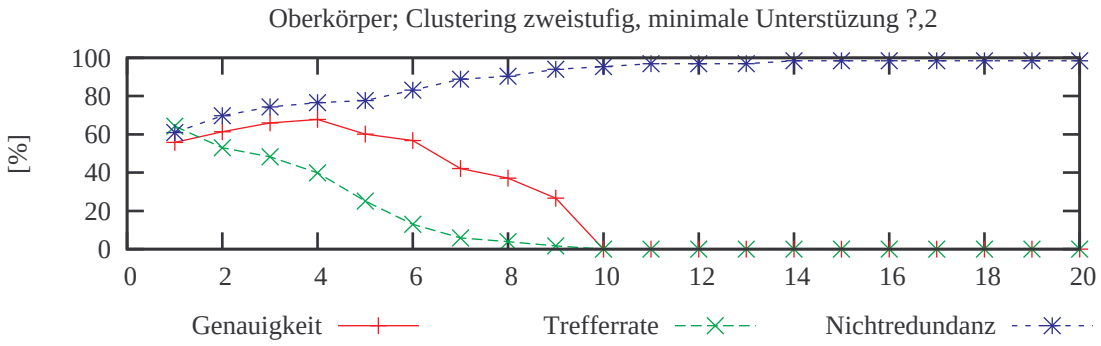


Abbildung 47: Evaluierung von Mean-Shift-Cluster Schwellwerten für Oberkörper

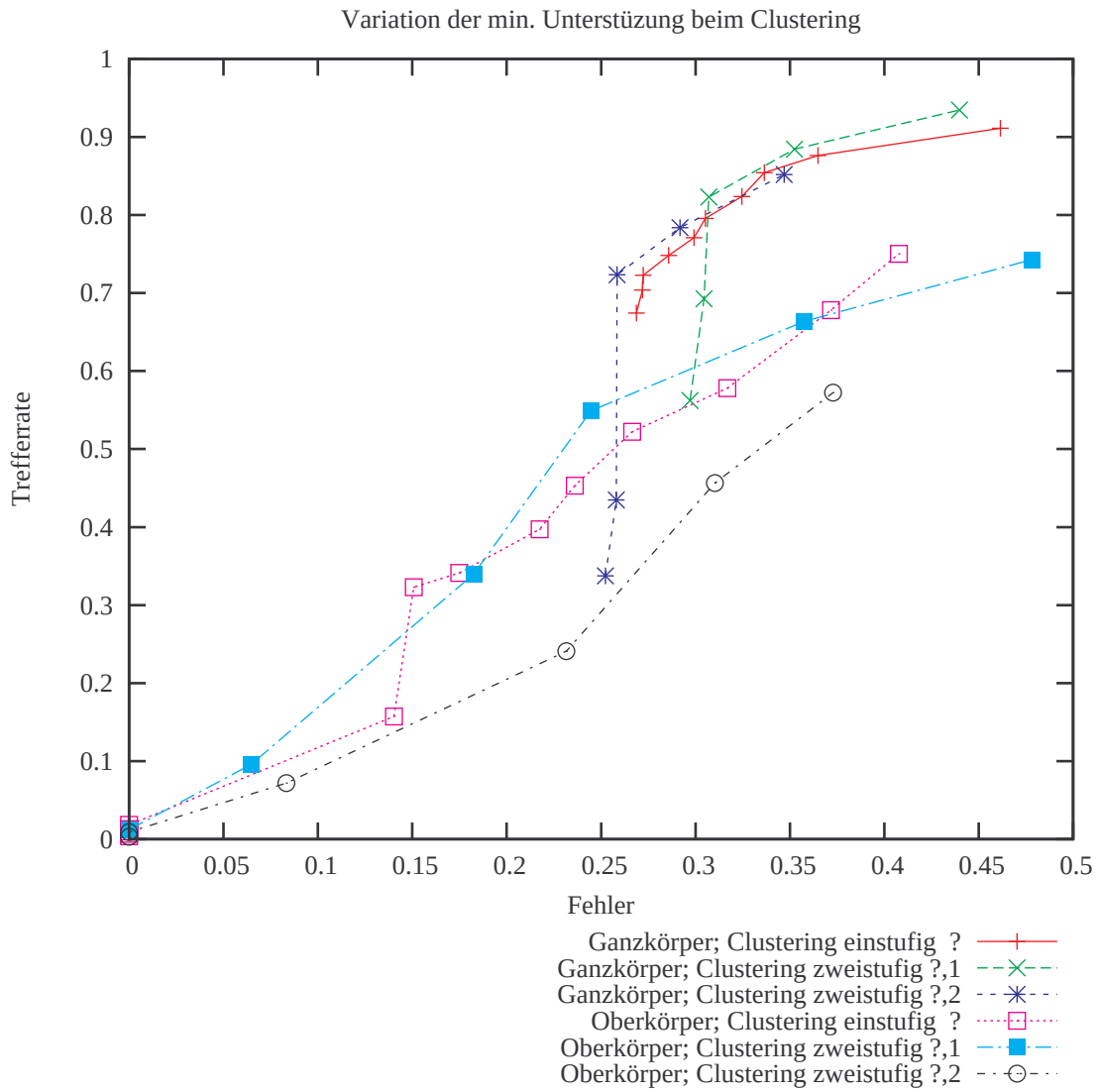


Abbildung 48: Qualität bei Variation der minimalen Unterstützung beim Clustering

5.2.6 Bewertung

Bei der Konfiguration der HOG-Detektoren für den Echtzeiteinsatz bleibt Raum für Verbesserungen. Dazu zählen die folgenden Punkte:

- Der Parametersatz für Oberkörper führt zu etlichen Fehldetektionen. Dies geht vermutlich darauf zurück, dass hier weniger markante Kanten vorliegen. Die Annotation von ganzen Umrissen führte zu sehr vielen Merkmalsausprägungen. Eine Einschränkung auf den Kopf-Schulter-Bereich würde hier sicherlich zu besseren Resultaten führen.

  Darüber hinaus erwies sich das gewählte feste Seitenverhältnis für beide Objekttypen als nicht immer passend, vor allem weil in Menschenmengen so die Körper der Nachbarn mit erfasst werden. Eine Verwendung von mehreren Stufen wäre hier passender.

- Bei dem gewählten groben räumlichen Suchraum zeigt das Clustering nicht ganz den gewünschten Effekt. Nicht umsonst werden in der Literatur um ein bis zwei Größenordnungen größere Suchräume gewählt. Die Wahrscheinlichkeit für eine Fehldetektion fällt mit größerer Sampledichte ab. Tests mit 30 statt 10 Tiefenebenen bei der Ganzkörperdetektion zeigten einen Gewinn von 10 bis 15 Prozentpunkten bei der Genauigkeit bei dreifacher Rechenzeit.

- Das Parameter-Tupel (Neuronenschwelle, Kernelbandbreite, Verschmelzungsschwelle, Schwelle, Support) beeinflusst die Entscheidung des Detektors, wobei alle Parameter des voneinander abhängig sind. Eine bessere und schnelle Strategie zum Finden des optimalen Parametersatzes wäre wünschenswert.

- Die insgesamt über 2.500 annotierten Kamerabilder aus 4 Szenarios stellen eine solide Grunddatenmenge dar. Die Bilder innerhalb einer Aufnahmeserie sind sich allerdings so ähnlich, das auch die Reduktion auf jedes zehnte Bild bis auf wenige Prozentpunkte ähnliche Evaluierungsergebnisse bringt. Statt der mit über 3 fps sehr dichten Aufnahmeserien wären – getreu dem Wahlspruch „there is no data like more data “ [38] – weitere Aufnahmereihen mit anderen Personen und Hintergründen eine Grundlage, die Detektoren robuster zu machen.

- Gezieltes Nachtrainieren mit Fehldetektionen [49] oder Ada Boosting können auch zur Verbesserung der Detektorqualität eingesetzt werden.[15]

Qualität

In guten Szenarios (etwa im weißen Raum neben der FINCA) erreichen beide Detektoren knapp über 75% Trefferrate. Der Oberkörperdetektor schafft im besten Fall über 80% Genauigkeit, während der Ganzkörperdetektor hier 95% erreicht (Abbildung 49). Schwierige Daten wie durch das Bild laufende, abgeschnittene Personen und gemusterter Hemden direkt vor der Kamera führen zu vielen Fehldetektionen. Der Ganzkörperdetektor bleibt dennoch in allen Testszenarios mit über 55% Trefferrate und 50% Genauigkeit einigermaßen verlässlich. Demgegenüber fällt der Oberkörperdetekor auf unter 50% Trefferrate ab. Die Mehrzahl der vorkommenden Fehldetektionen fallen in eine von zwei Kategorien. Beide führen zu unpassenden hohen Entfernungswerten. Die erste wird von Objekten direkt vor der Kamera ausgelöst, etwa die

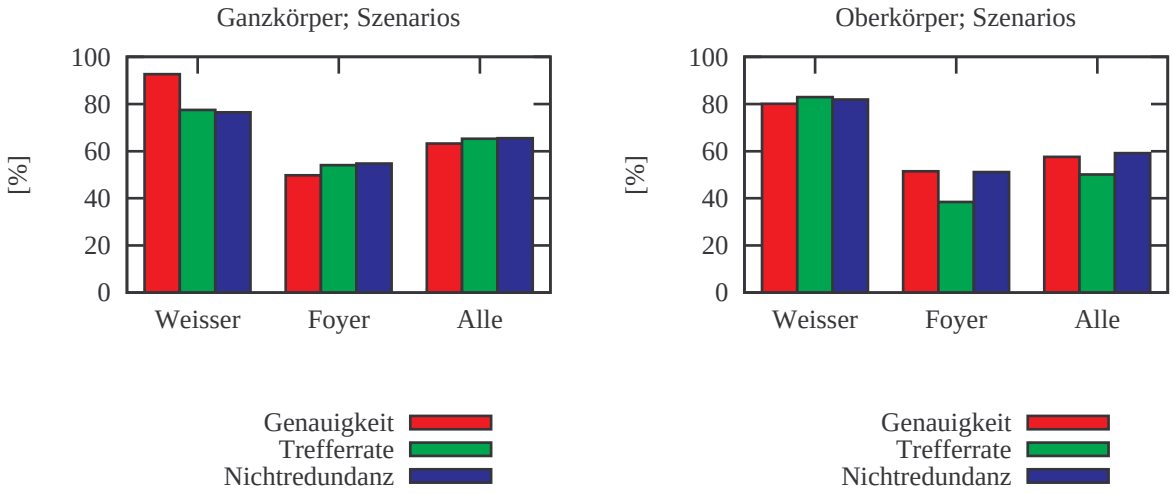
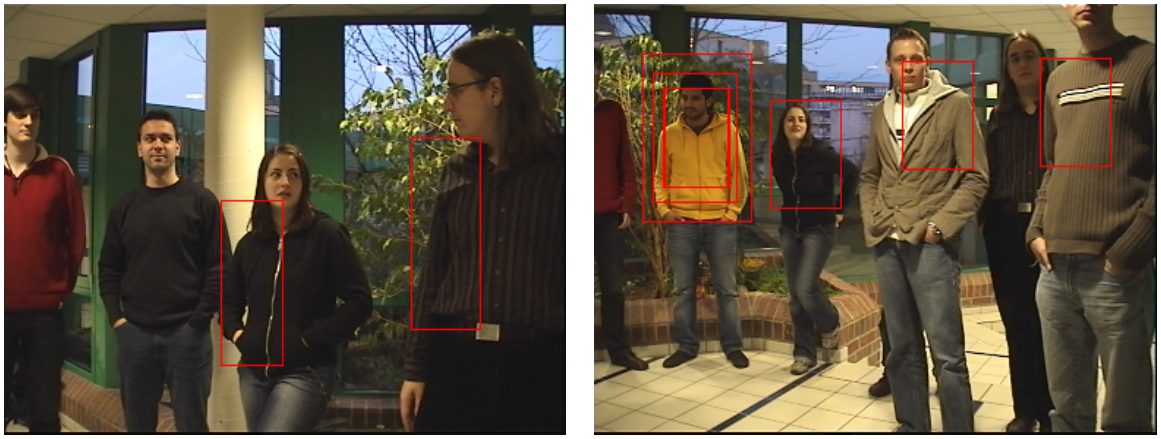


Abbildung 49: Ergebnisse der HOG Basierten Detektoren nach Optimierung der Parameter



(a) Ganzkörper (b) Oberkörper

Abbildung 50: Typische Fehldetektionen, die zu einer falschen Entfernungs- und Winkelangabe führen

Detektion von Oberkörpern in gemusterten Hemden. Die zweite Gruppe werden Fehldetektionen aufgrund von gemusterten Wänden oder Strukturen in Wandnähe gebildet. Da vielfach sehr kleine Detektorfenster bei Fehldetektionen beteiligt sind, lässt sich deren Zahl durch eine Einschränkung des Entfernungsbereiches von 12m auf 8m reduzieren. Zu beiden Typen von Fehldetektionen gibt es mit sehr hoher Wahrscheinlichkeit keine passenden Detektionen des Beinpaardetektors. Da diese in den meisten Fällen bei der Integration vorausgesetzt werden, ist nicht damit zu Rechnen, dass der PARTYBOT auf solche Fehldetektionen der HOG-Detektoren reagiert.

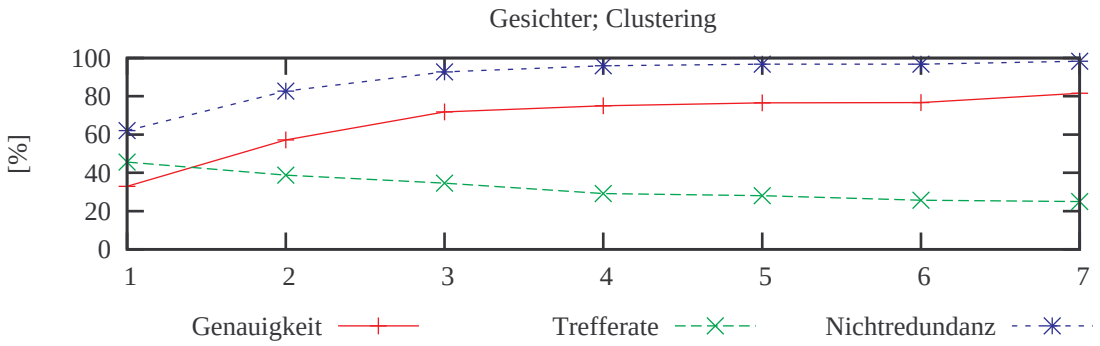


Abbildung 51: Evaluierung von Cluster Schwellwerten für Gesichter

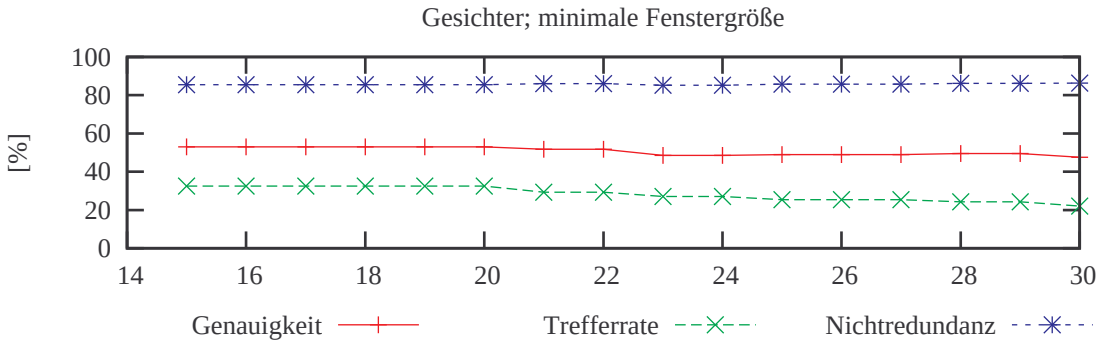


Abbildung 52: Evaluierung von minimale Größe der Gesichter

5.3 Evaluierung der Gesichtsdetektion

Einige Tests wurden basierend auf den Oberkörper-Annotationen für Testaufnahmen durchgeführt. Die Bewertung erfolgt analog zu den HOG-Detektoren. Die Bilder wurden nicht neu annotiert, statt dessen wurden Gesichtsdetektionen, welche zu 90% mit dem oberen Drittel einer Oberkörperannotation überlappten, als richtige Detektionen gezählt. Diese Strategie war mehr als hinreichend, da für keines der Bilder eine Detektion falsch klassifiziert wurde.
Da die Gesichtsdetektion lediglich aus einer Bibliothek eingebunden wurde, mussten wir nur wenige Parameter anpassen. Ausgehend von derselben Datenbasis konnten Vergleiche aller bildbasierten Detektoren angestellt werden.

5.3.1 Parameteroptimierung

Für die Gesichtsdetektion wurde lediglich der Parameter *minimale Unterstützung* (englisch *min. support*) des Clusterings optimiert. Wie in Abbildung 51 zu erkennen ist, werden bei einem Wert von 2-3 gerade noch viele Gesichter bei hinreichender Genauigkeit gefunden. In keinem Szenario war der Wert drei überlegen, sodass für den PARTYBOT der Wert zwei als Einstellung übernommen wurde. Die Rechenzeit steigt mit dem Schwellwert und liegt hier bei etwa 300ms pro Bild.

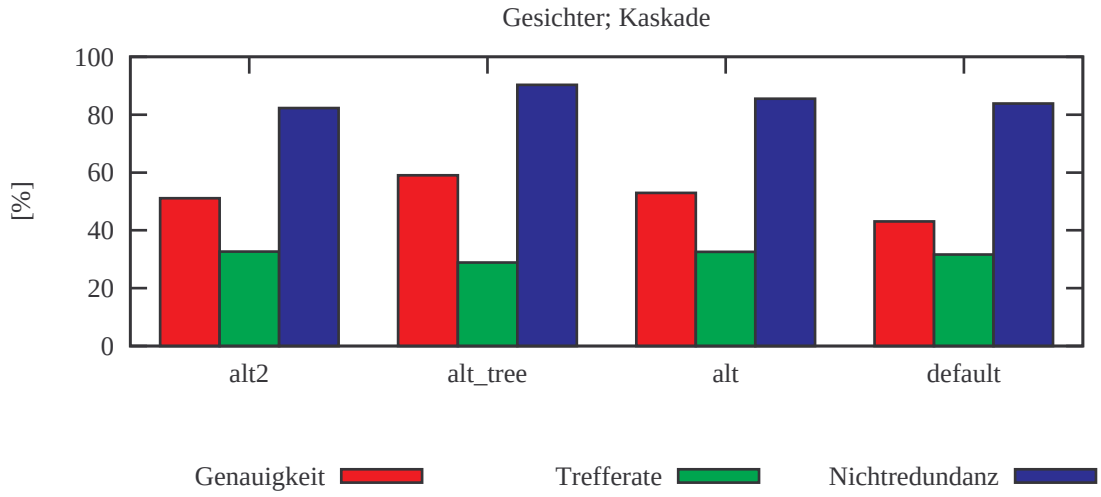


Abbildung 53: Evaluierung von verschiedenen Haarkaskaden für Gesichter

Die minimale Gesichtsgröße lässt sich direkt aus der Anwendungsvorgabe berechnen. Bei einer maximalen Personenentfernung von 12m ergibt sich mit dem Blickwinkel $\gamma = 18°$ auf Basis einer durchschnittlichen Gesichtshöhe von $\approx$ 0,21m eine Höhe von über 15 Pixeln (Gleichung 30).

$$\text{(30)}\quad \text{Fensterhöhe [Pixel]} = \frac{\text{Bildhöhe [Pixel]} \cdot \text{Gesichtshöhe [m]}}{\text{Abstand [m]} \cdot \tan\gamma} = \frac{288 \cdot 0,21}{12 \cdot \tan 18°} \geq 15$$

Evaluierungen mit Testaufnahmen zeigen, dass auch für 20 Pixel nicht weniger Gesichter gefunden werden (siehe Abbildung 52). Damit beträgt die effektive maximale Entfernung $d_{\max}$, in der Gesichter gefunden werden, 9,31m.

5.3.2 Kaskade

Ein Vergleich der verschiedenen Detektorkaskaden ergab eine gute Eignung von den Alternativkaskaden `alt` und `alt2` mit etwa 50% Genauigkeit und knapp 30% Trefferrate der vorkommenden Gesichter. Dahingegen findet `alt_tree` weniger Gesichter findet und die Standardkaskade `default` ist 39% durchschnittlicher Genauigkeit deutlich schlechter (Abbildung 53). Die Zahlen wurden auf der Gesamtmenge der Bilder ermittelt und umfassen die im Folgenden getrennt evaluierten Szenarien.

5.3.3 Bewertung

Die Gesichtserkennung findet in unter einer halben Sekunde aufrechte, frontale Gesichter. Ein in der Kameraebene gedrehtes Gesicht (englisch *in-plane rotation*) oder ein von der Kamera weggeneigter Kopf wird nicht erkannt. Einige der auftretenden Fehldetektionen wären durch einen Farbraumtest vermeidbar gewesen (Schilder, Türklinken usw.).
Die Winkelbestimmung ist naturgemäß recht exakt. Die Entfernungsschätzung ist erstaunlich genau, was vermutlich auf gering variierende Gesichtshöhen zurückzuführen ist. Eine typische Fehldetektion, bei der

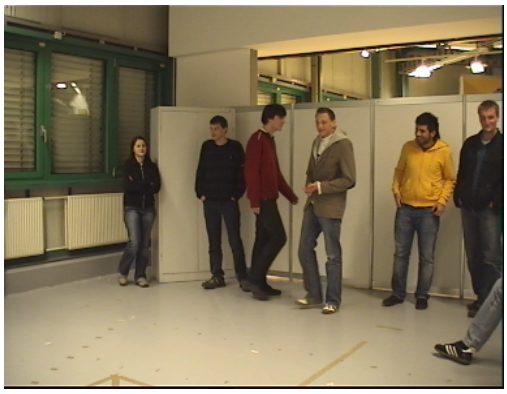


Abbildung 54: Szenario „WhiteRoom“

Abbildung 55: Szenario „Foyer“

ein Kopf als „Mund “ eines Gesichtes detektiert wird, führt zu einem etwa 2,5fach zu kleinen Abstand, während die anderen Werte passend bleiben. Dies tritt in ca. 1% der Fälle auf.

5.4 Evaluierung der kamerabasierten Detektoren im Vergleich

Auf der Menge der für das Training und Evaluieren der HOG-Detektoren annotierten Testaufnahmen wurden gezielt Vergleichstests durchgeführt, um die Eignung der einzelnen Detektoren für bestimmte Anwendungsfälle vergleichen zu können. Die Skripte zur Aufnahme und Auswertung der Detektorergebnisse der verschiedenen bildbasierten Detektoren wurden dazu passend zusammengeführt, so dass nicht nur die HOG-basierten Detektoren untereinander, sondern diese auch mit dem Viola-Jones basierten Gesichtsdetektor verglichen werden konnten. Die HOG-Detektoren verwendeten die von Kamerasteuerung bekannten Aufnahmewinkel zur Schätzung der Fußbodenposition.

5.4.1 Testszenarios

Aus der Menge der annotierten Aufnahmen wurden zwei Abschnitte ausgewählt, welche sich in der Entfernung der Personen signifikant unterscheiden. Wie sich im Laufe der Auswertungen zeigte, ist die Verwendung jedes zehnten Bildes der ursprünglich kontinuierlich aufgenommenen Bilder hinreichend repräsentativ, da die Ergebnisse meist um weniger als 2 Prozentpunkte abweichen.

Szenario 1: WhiteRoom

Das Szenario „WhiteRoom“ (Abbildung 54) enthält vorwiegend Aufnahmen weit entfernter und getrennt voneinander stehender Personen vor weißem Hintergrund. Die Entfernung der Personen beträgt zwischen 4 und 8m. Sie stehen meist getrennt vor weitgehend ungemustertem Hintergrund. Fußboden und Füße sind quasi durchgehend im Bild.

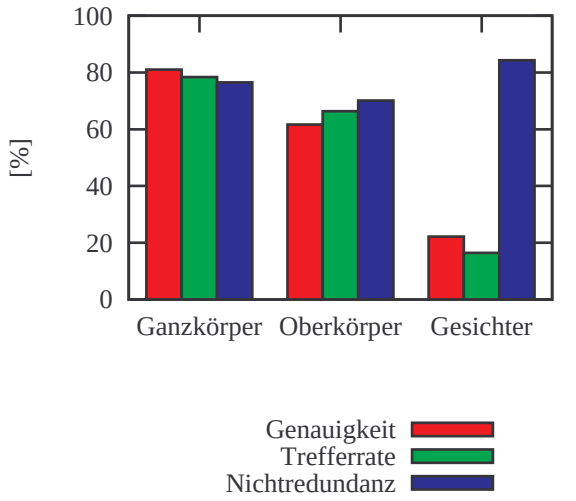


Abbildung 56: Detektionen „WhiteRoom“

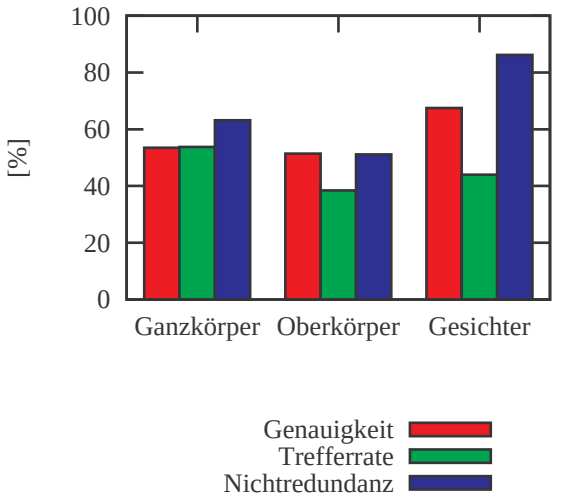


Abbildung 57: Detektionen „Foyer“

Szenario 2: Foyer

Das Szenario „Foyer“ (Abbildung 55) enthält vorwiegend Aufnahmen naher, durcheinanderstehender Personen vor abwechslungsreichem Hintergrund. Die Entfernung der Personen beträgt zwischen 2 und 4m. Die Füße der Personen sind nicht immer mit im Bild. Die Personen stehen eng beieinander und teilweise voreinander vor der Kamera.

5.4.2 Ergebnisse

Abhängig von der Situation erweisen sich die Detektoren als mehr oder weniger geeignet. Im ersten Szenario (Abbildung 56) erweist sich der Ganzkörperdetektor mit einer Genauigkeit von 75% als zuverlässig. Mit 76% mittlerer Trefferrate werden sehr häufig Personen gefunden. Demgegenüber treten im zweiten Szenario mehr Fehldetektionen auf. Der Oberkörper und Ganzkörperdetektor entdecken im ersten Szenario über zwei Drittel der annotierten Personen. Im Foyer wird immerhin noch ein Drittel gefunden. Der Gesichtsdetektor ist im zweiten Szenario mit einer Genauigkeit von 68% der mit Abstand zuverlässigste Detektor, (Abbildung 57). Mit einer Trefferrate von 44% stellt er hier auch beinahe so viele Detektionen wie der Personenerkenner. Demgegenüber werden im ersten Szenario in erster Linie scheinbare Gesichter gefunden, mit einer Trefferrate von 15% und einer Genauigkeit von 23% ist hier der Einsatz des Gesichtserkenners nicht mehr sinnvoll.

5.4.3 Bewertung

Die Unterschiede sind vermutlich hauptsächlich auf die Entfernung zurückführen. So erweisen sich der Oberkörper- und Gesichtsdetektor in dem „Foyer“ als effektiver, da selten ganze Personen im Bild sind. Demgegenüber kann der Ganzkörperdetektor im „WhiteRoom“ klar überzeugen. Die insgesamt geringe Trefferrate ist darauf zurückzuführen, dass viel Personen stark von der trainierten Pose abweichen. Der

Detektor	Ganzkörper	Oberkörper	Gesichter
Genauigkeit der Detektionen je Bild [%]	62,59±46,6	60,01± 44,3	23,26±39,8
Trefferrate der Personen je Bild [%]	76,69±19,2	63,44± 36,5	16,41±34,5
Nichtredundanz je Bild [%]	76,52±18,0	66,09± 35,1	84,34±35,9
Korrekte Detektionen (TP)	183	237	18
Fehldetektionen (FP)	60	117	63
Genauigkeit insgesamt [%]	75,31	66,95	22,22
Zeit pro Bild [ms]	691± 237	224,86±138	248± 104

Tabelle 7: Test der bildbasierten Detektoren im Szenario „WhiteRoom“

Detektor	Ganzkörper	Oberkörper	Gesichter
Genauigkeit der Detektionen je Bild [%]	43,58±40,2	48,89± 33,9	67,51±33,4
Trefferrate der Personen je Bild [%]	54,06±33,6	39,97± 32,8	43,96±37,2
Nichtredundanz je Bild [%]	54,80±30,0	49,98± 32,4	86,18±26,5
Korrekte Detektionen (TP)	113	148	116
Fehldetektionen (FP)	114	143	56
Genauigkeit insgesamt [%]	49,78	50,86	67,44
Zeit [ms]	763± 234	296,39±168	178± 87

Tabelle 8: Test der bildbasierten Detektoren im Szenario „Foyer“

Oberkörperdetektor zeigt im ersten Szenario, vermutlich wegen der eindeutigeren Bildsituationen, auch bessere Ergebnisse. Der Gesichtdetektor ist hier klar im Nachteil, allein weil die Auflösung der Gesichter die Grenze des Nutzbaren unterschreitet. Insgesamt ist die geringe Trefferrate des Gesichtsdetektors auch daraus erklärbar, dass dieser nur auf frontale, aufrechte Gesichter reagiert.
In Anbetracht der hohen Abhängigkeit von Entfernung und Qualität der Detektoren ist es sinnvoll, Gesichtsdetektionen nur unter 4m zuzulassen. Ganzkörperdetektionen sind nur in über 3-4m Entfernung sinnvoll, da sonst der Fußboden nicht im Bild ist. Die Oberkörperdetektionen können in allen Fällen als Hinweis auf eine Person gewertet werden, wegen der niedrigen Genauigkeit ist aber darauf zu achten, dass diese nur mit einer zweiten Detektion zusammen als Auslöser für Handlungen des PARTYBOT genommen werden. Vorzugsweise sollte diesen nur mit einer Detektion eines anderen Detektors oder einer zweiten Detektion nach Bewegung der Kamera Vertrauen geschenkt werden. Diese Strategie ist insgesamt hilfreich. So verifiziert der PARTYBOT grundsätzlich die Detektionsergebnisse noch einmal mit einem zweiten „Hinsehen“.

5.5 Evaluierung des Beindetektors

Ziel der Evaluierung des Beindetektors ist es, die bestmöglichen Werte für die Parameter des Beindetektors zu bestimmen und die Funktionalität zu validieren. Dazu werden zwei Szenarien überprüft.

5.5.1 Szenarien

Zum einen das in Abbildung 58 dargestellte Szenario 1, in dem alle Objekte und Personen statisch sind. Hier werden die Parameter in allen möglichen Kombinationen gemäß Tabelle 9 kombiniert und jede Kombination fünf mal verwendet, um einen Mittelwert bilden zu können. Daraus ergeben sich insgesamt 2625 Durchläufe

des Beindetektors. Mit einer Laufzeit von etwa 315s auf dem Roboter benötigt jeder Durchlauf somit 0,12s. Beim Szenario 2 wurden im selben Raum Werte aufgezeichnet, wobei die Testpersonen sich durch den Raum bewegen. Hier werden 500 Durchläufe aufgenommen. Die Parameter werden hier gemäß der zweiten Zeile der Tabelle 9 verwendet und entsprechen dabei den Werten, die in [29, S.64 ff] verwendet wurden.

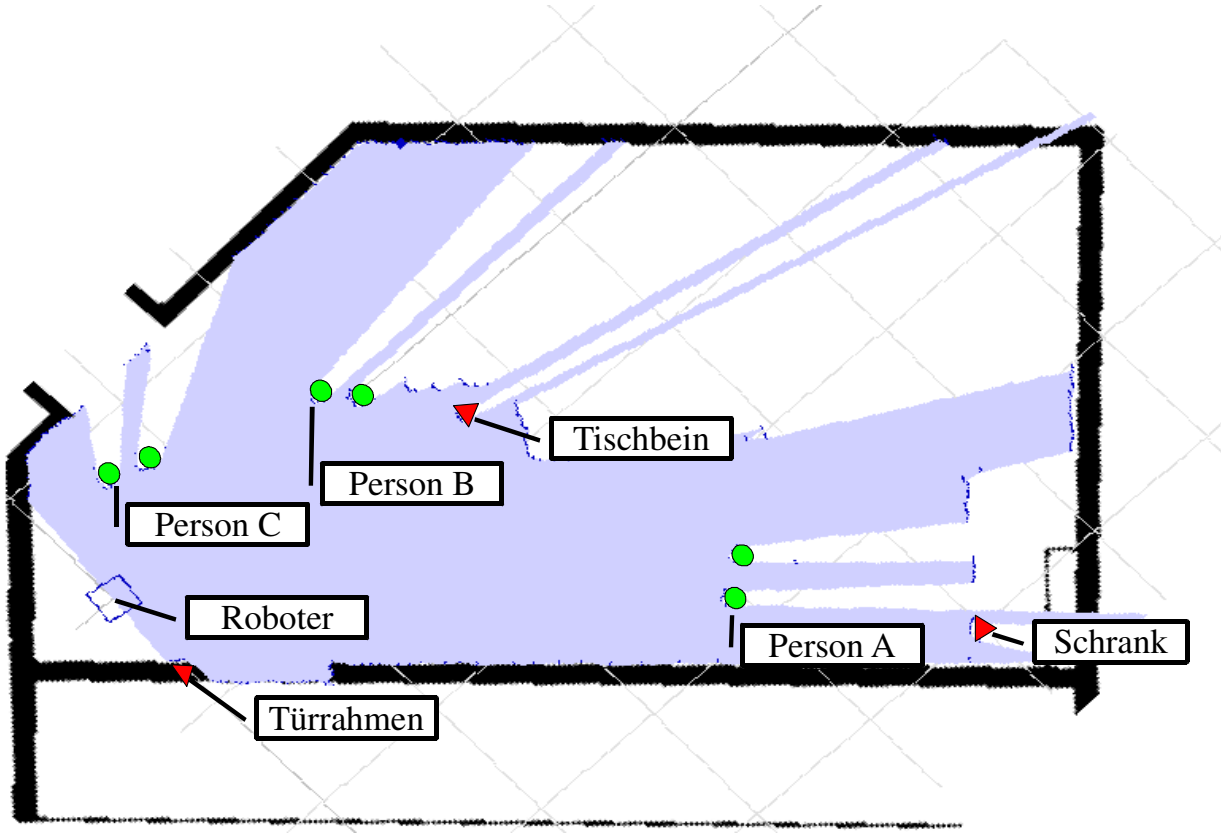


Abbildung 58: Evaluierungsszenario 1 für den Beindetektor

Szenario	Parameter 1 min. Anzahl der Elemente pro Segment	Parameter 2 min. Beinbreite [cm]	Parameter 3 max. Beinbreite [cm]	Parameter 4 max. Beinabstand [cm]
1	3 - 5 (1)	4 - 10 (1)	23 - 27 (1)	40 - 60 (5)
2	5	5	25	50

Tabelle 9: untersuchte Parameter, Schrittweite in Klammern

5.5.2 Daten

Die Ergebnisse des ersten Szenarios sind in den nachfolgenden Tabellen dargestellt. Die Tabelle 10 zeigt wie oft welches Objekt detektiert wurde. In Tabelle 11 werden nur die Testpersonen betrachtet, die anschließend in Tabelle 13 einzeln betrachtet werden bezüglich der verschiedenen Kombinationen der Parameter 1 und 4. In Tabelle 11 ist zu sehen, dass die beiden Personen, die dem Roboter am nächsten sind, immer bzw. fast immer erkannt werden. Person A, die etwa 4m entfernt steht, wird hingegen nur in 67% der Durchläufe erkannt. Zum einem sieht man hier, dass die Wahrscheinlichkeit detektiert zu werden von der Entfernung abhängt, was in direkten Zusammenhang mit der minimalen Anzahl von Elementen pro Segment steht. In Tabelle 13 wird deutlich, dass eine Mindestanzahl von fünf Elementen zu einer schlechteren Detektion in 4m Entfernung führt. Dieser Zusammenhang wird noch einmal bei der Betrachtung von Tabelle 12 deutlich. Zum anderen kann man in Tabelle 10 erkennen, dass die Laserdaten auch mit einem gewissen Fehler behaftet sind. Selbst die Entfernung glatter, statischer Objekte, wie dem Türrahmen, dem Schrank und dem Tischbein, variiert. Ungleichmäßige und sich leicht bewegende Oberflächen, wie Beine in einer Jeanshose, verursachen dementsprechend größere Abweichungen. Diesen Effekt verdeutlicht ein direkter Vergleich der Werte für Person B und C in Tabelle 13. Die Person, die näher zum Roboter steht, wird hier nicht immer erkannt, während die andere immer erkannt wird. Ursachen sind hier in den leichten Bewegungen der Testperson und dem Fehler des Lasers zu suchen. Die Änderungen im Laserprofil des Hosenbeines durch Verformungen und Faltenwurf spielen hier eine entscheidende Rolle.

Objekt	Winkel	detektierte Entfernung [cm]	Detektionen
Türrahmen	4,5°	61,5±1,5	2625
Schrank	49°-50°	558,0±1,0	220
Person A (rechtes Bein)	51°-53°	396,5±2,5	406
Person A (beide Beine)	54°-55°	397,5±1,5	1069
Person A (linkes Bein)	55°-57°	399,0±4,0	290
Tischbein	82°-83°	258,0±1,0	48
Person B (beide Beine)	97°-99°	203,0±1,0	2590
Person B (linkes Bein)	100°-101°	193,5±1,5	35
Person C (rechtes Bein)	125°-128°	92,0±1,0	31
Person C (beide Beine)	132°-135°	83,5±0,5	2256
Person C (linkes Bein)	140°-143°	74,0±1,0	323

Tabelle 10: Die in Szenario 1 detektierten Objekte, mit Winkel, Entfernung und Anzahl der Detektionen bei 2625 Durchläufen, sind hier dargestellt. Die Anzahl der Gesamtdetektionen einer Person ergibt sich aus der Summe der Detektionen für das linke und rechte Bein, sowie der Detektionen beider Beine.

Aus Abbildung 59 kann man entnehmen, dass eine Mindestanzahl von fünf Elementen je Segment zu weniger Fehldetektionen, aber auch zu weniger richtigen Detektionen führt, als eine Mindestanzahl von drei oder vier. Es gibt keinen signifikanten Unterschied zwischen einer Mindestanzahl von drei oder vier Elementen je Segment, was auf das Szenario zurückzuführen ist. Eine Variation des maximalen Beinabstandes wirkt sich hier kaum auf die Anzahl der Detektionen aus. Deutlich werden diese Beobachtungen noch einmal in Tabelle 15. Die Precision-Werte in der zweiten Zeile werden jedoch leicht verzerrt durch die Tatsache, dass der Türrahmen in allen Durchläufen als Bein detektiert wird. Keine Konfiguration der Parameter kann diese

Person	rechtes Bein	beide Beine	linkes Bein	Gesamt	Recall
A	406	1069	290	1765	67,2
B	0	2590	35	2625	100,0
C	31	2256	323	2610	99,4
Gesamt	437	5915	648	7000	88,9

Tabelle 11: detektierte Personen mit Anzahl der Detektionen und Recall bei 2625 Durchläufen

Entfernung [m]	3 El. (1,5°) [cm]	4 El. (2,0°) [cm]	5 El. (2,5°) [cm]
1	2,6	3,5	4,4
2	5,2	7,0	8,7
3	7,86	10,5	13,1
4	10,5	14,0	17,4
5	13,1	17,5	21,8

Tabelle 12: Die Mindestbreiten, die ein Bein rein rechnerisch besitzen muss, um mit der entsprechenden Anzahl von Elementen in der angegeben Entfernung wahrgenommen werden zu können.

	Person A			Person B			Person C		
Beinabstand[m]	3 El.	4 El.	5 El.	3 El.	4 El.	5 El.	3 El.	4 El.	5 El.
0,40	172	167	13	175	175	175	173	175	175
0,45	173	165	5	175	175	175	174	174	175
0,50	172	165	20	175	175	175	172	175	175
0,55	173	168	16	175	175	175	171	175	174
0,60	172	164	20	175	175	175	172	175	175
Gesamt	862	829	74	875	875	875	863	874	874
Recall [%]	98,5	94,7	8,5	100,0	100,0	100,0	98,6	99,9	99,9

Tabelle 13: Detektionen der Personen bei 175 Durchläufen für jede Kombination der Parameter 1 und 4

	Türrahmen			Schrank			Tischbein		
Beinabstand[m]	3 El.	4 El.	5 El.	3 El.	4 El.	5 El.	3 El.	4 El.	5 El.
0,40	175	175	175	26	20	0	3	5	0
0,45	175	175	175	20	22	0	3	7	0
0,50	175	175	175	22	20	0	4	7	0
0,55	175	175	175	23	23	0	6	5	0
0,60	175	175	175	24	20	0	6	2	0
Gesamt	875	875	875	115	105	0	22	26	0

Tabelle 14: Detektionen der Nichtpersonen bei 175 Durchläufen für jede Kombination der Parameter 1 und 4

	3 Elemente	4 Elemente	5 Elemente	Gesamt
Recall	99,1	98,2	69,5	88,9
Precision	72,0	71,9	67,6	70,8
Precision ohne Türrahmen	95,0	95,2	100,0	96,3

Tabelle 15: Precision und Recall des ersten Evaluierungsszenarios im Vergleich

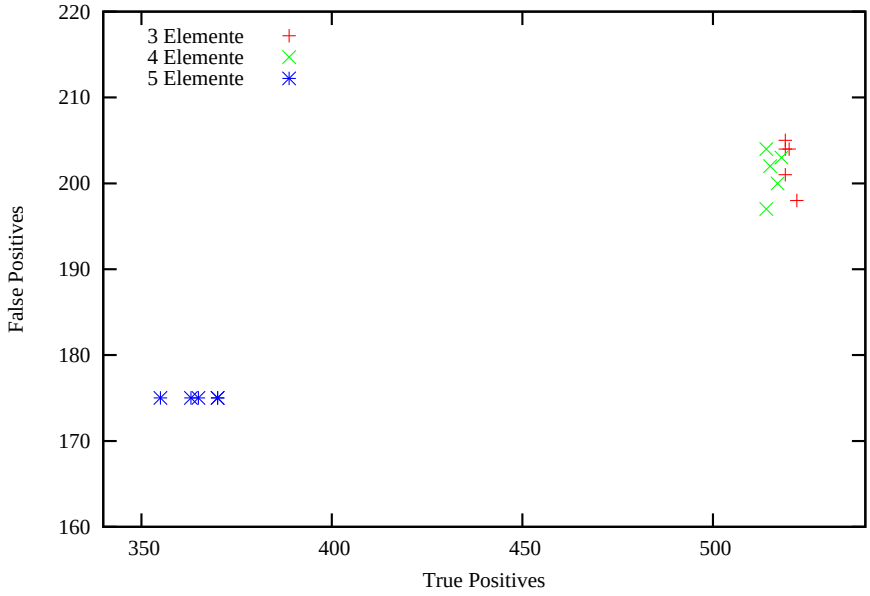
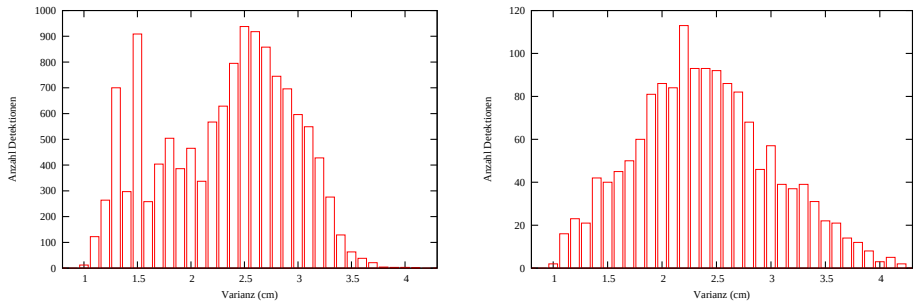


Abbildung 59: Hier ist eine grafische Zusammenfassung der Tabellen 13 und 14 gegeben. Aufgetragen sind die Detektionen für alle 15 Kombinationen der Parameter 1 und 4.

Fehldetektion ausschließen, da sowohl die Varianz vom mittleren Abstandes des Segmentes vom Roboter, als auch die Breite der von menschlichen Beinen entsprechen. Schließt man diese Detektionen in der Auswertung aus, so erkennt man, dass bei fünf Elementen ausschließlich Personen detektiert werden. Deutlich wird dies zudem in Tabelle 14.

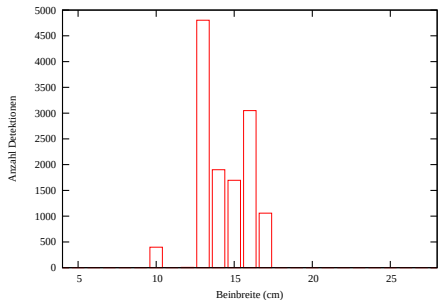
Da die Variation des maximalen Beinabstandes keine erkennbaren Verbesserungen bringt, wird der in [29, S.64 ff] benutzte Abstand von 50cm verwendet. Bei der Mindestanzahl von Elementen je Segment wird ebenfalls der dort benutzte Wert von fünf Elementen beibehalten. Diese Wahl hat zur Folge, dass in der Regel weniger richtige Detektionen, aber auch weit weniger Fehldetektionen auftreten. Die Entscheidung wird zugunsten einer höheren Zuverlässigkeit getroffen. Spezialfälle wie die Detektion des Türrahmens lassen sich nicht ausschließen, sie erhalten jedoch durch die Güteberechnung (siehe Gleichung 5) eine niedrigere Priorität. Eine weitere Einschränkung dieser Wahl ist der kleinere Bereich, in dem Personen detektiert werden können. Personen, die weiter als 3m vom Roboter entfernt stehen, werden von weniger Laserstrahlen erfasst und werden in Folge dessen seltener detektiert.

Zur Festlegung der letzten vier Parameter werden beide am Anfang vorgestellten Szenarien betrachtet. Die Intervalle, in der die Varianz vom mittleren Abstand der Segmente zum Roboter und die Beinbreite liegen dürfen, sollen eingegrenzt werden. Wir betrachten nur Objekte als mögliche Beine, die eine Varianz von 1 bis 4,2cm vom mittleren Abstandes des Segmentes vom Roboter besitzen. Die Histogramme in Abbildung 60(a) und 60(b) enthalten keine Werte außerhalb dieses Bereiches. Anhand der in Abbildung 60(c) und 60(d) dargestellten Ergebnisse haben wir uns bei der Beinbreite für ein Intervall von 8 bis 25cm entschieden.

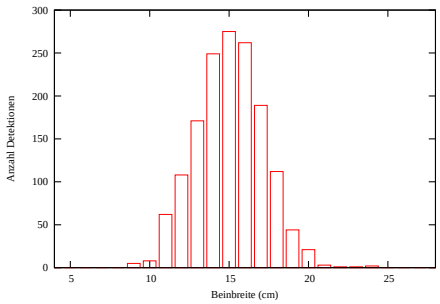


(a) Varianzen Szenario 1 (statisch)

(b) Varianzen Szenario 2 (dynamisch)



(c) Beinbreiten Szenario 1 (statisch)



(d) Beinbreiten Szenario 2 (dynamisch)

Abbildung 60: berechnete Varianzen gegenüber dem mittleren Abstand der Segmente vom Roboter und detektierte Beinbreiten in den beiden Szenarien

5.5.3 Bewertung

Das implementierte Verfahren findet in der Regel alle nicht verdeckten Beine in der näheren Umgebung bis 3m. Verdeckte Beine werden zum Teil erkannt, wenn sie noch ausreichend sichtbar sind. Der Algorithmus arbeitet effizient und benötigt ca. 0,12s auf dem Roboter und bietet damit vor allem Vorteile gegenüber den anderen verwendeten Detektoren. Ab einer Entfernung von 3m lässt die Zuverlässigkeit der Detektion nach und es werden nicht mehr alle Beine entdeckt. Schwächen weist auch die Gruppierung auf, da wir an dieser Stelle kein Tracking oder Bewegungsmodell einsetzen. So können Beine falsch gruppiert werden. Um diesem Effekt entgegenzuwirken, wird jeder detektierten Person eine Güte (siehe Gleichung 5) zugeordnet.

5.6 Evaluierung der Detektoren mit Kamerabewegung

In einem integrierten Test wurden alle vier Detektoren zusammen mit der durch Beinpaardetektionen gesteuerten Kamerabewegung evaluiert. Ziel war es, die Genauigkeit der Positionsschätzung und die Trefferrate der Detektoren zu prüfen. Dabei wurde bewusst ein Nahbereichszenario gewählt, in dem der Beinpaardetektor und Gesichtsdetektor gute Voraussetzungen haben. Die Funktion der HOG-Basierten Detektoren zum Aufspüren von Personen in Entfernungen von über 4m wurde in den vorhergehenden Abschnitten hinreichend ausführlich getestet.

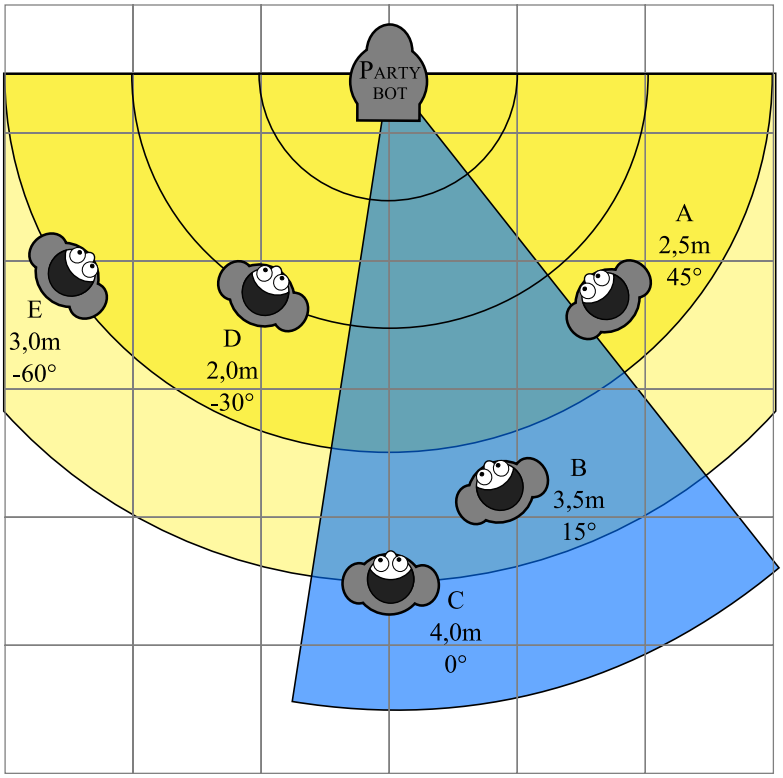


Abbildung 61: Detektorevaluierungsszenario – Der Sichtbereich der Kamera ist für das Zentrieren auf Person B nach Detektion durch den Beinpaarerkenner eingezeichnet.

5.6.1 Szenario

Dabei haben sich fünf Personen vor dem PARTYBOT in einem Halbkreis aufgestellt. Die Personen nahmen in -45, -15, 0, 30 und 60° Entfernungen zwischen 2 und 4m ein und blickten zum PARTYBOT, wie in Abbildung 61 gezeigt. Die Kamera schwenkte dabei zufällig zu Beinpaardetektionen und ließ so den Sichtkegel wandern. Person C befand sich dabei außerhalb der vom Beinpaardetektor gut erfassten Entfernung, wurde aber teilweise beim Fokussieren von Person B miterfasst.

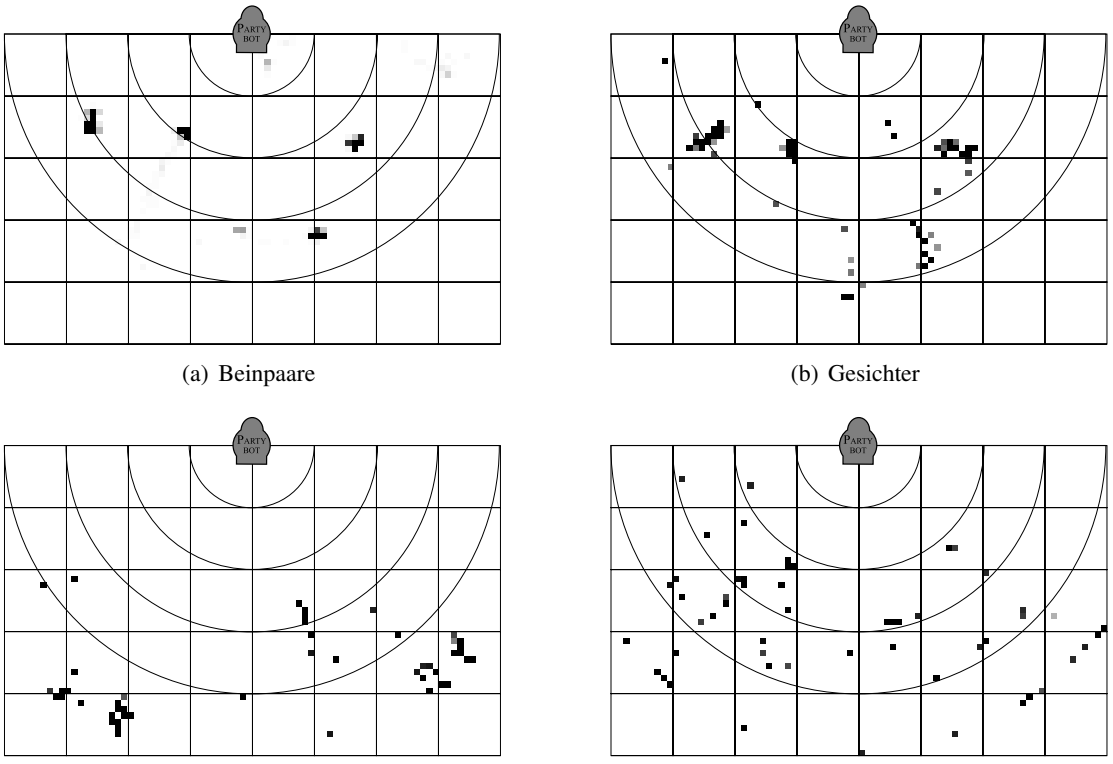
5.6.2 Daten

Die Detektionen wurden über einige Minuten aufgezeichnet. Nach einer Segmentierung nach Winkelbereichen ergeben sich die Detektionsanzahlen nach Tabelle 16. Die weit entfernte Person C wird sehr selten detektiert, Person D häufiger als alle anderen. Der Beinpaardetektor stellt mit 40% die meisten Detektionen. Die Detektionen aufgrund des Kamerabildes stellen etwa 20% für alle drei Detektoren.

Die Detektionspositionen werden in Abbildung 62 dargestellt. Der Beinpaardetektor lokalisiert die Personen bis auf Person C recht klar. Bei dem Plot für den Gesichtsdetektor ist auch diese zu sehen, hier ist die Streuung in Entfernungsrichtung größer. Der Ganzkörperdetektor lokalisiert die Personen etwas zu weit hinten und mit noch größerer Streuung. Beim Oberkörperdetektor ist die Streuung sehr groß, so dass sich die Detektionspositionen zu überschneiden beginnen.

	Beinpaare		Gesicht		Oberkörper		Ganzkörper		
	Anzahl	[%]	Anzahl	[%]	Anzahl	[%]	Anzahl	[%]	Summe
Person A	1.037	34,18	592	19,51	815	26,86	590	19,45	3.034
Person B	2.545	53,60	706	14,87	829	17,46	668	14,07	4.748
Person C	57	21,76	119	45,42	57	21,76	29	11,07	262
Person D	2.831	27,77	1.397	13,70	3.433	33,67	2.535	24,86	10.196
Person E	2.429	56,32	1.488	34,50	321	7,44	75	1,74	4.313
–	47	13,86	83	24,48	209	61,65	0	0,00	339
Gesamt	8.946	39,08	4.385	19,16	5.664	24,74	3.897	17,02	22.892

Tabelle 16: Detektionsanzahlen und Anteil der einzelnen Detektoren pro Person



(a) Beinpaare (b) Gesichter

(c) Ganzkörper (d) Oberkörper

Abbildung 62: Detektionsanzahlen der vier Detektoren je Raumposition. Jedes Kästchen entspricht 10x10cm in in einer Fläche von 8x5m vor dem PARTYBOT, je dunkler desto mehr Detektionen sind an dieser Stelle aufgenommen worden. Beim Ganzkörper- und Oberkörperdetektor zeigt sich im Vergleich zu den anderen Detektoren eine schlechte Verlässlichkeit der Entfernungsschätzung.

	A	B	C	D	E
Beinpaare	-43 ±1,2	-20 ±1,2	4 ±2,7	35 ±0,7	61 ±2,0
Gesicht	-40 ±2,5	-17 ±1,6	3 ±1,1	31 ±1,1	55 ±0,8
Oberkörper	-45 ±8,4	-10 ±2,9	1 ±1,0	36 ±5,9	54 ±5,8
Ganzkörper	-41 ±4,1	-17 ±1,0	2 ±0,0	32 ±5,5	55 ±2,0

Tabelle 17: Winkel der Detektionen in Grad je Person

	D	A	E	B	C
Beinpaare	194,13± 15,2	243,66±11,3	293,91± 5,6	339,96±32,2	319,25± 5,5
Gesicht	216,73± 11,4	242,25±14,8	290,53± 19,6	326,80±66,2	394,53±37,5
Oberkörper	354,24±111,8	429,06±97,7	394,35±106,0	309,69±39,1	395,16±78,0
Ganzkörper	491,50± 15,9	462,58±39,7	379,68± 23,3	297,50±45,5	406,00± 0,0

Tabelle 18: Entfernungswerte der Detektionen in cm je Person

Die Entfernungswert sind in Tabelle 18 aufgeführt. Die Entfernungsschätzung der Laserdaten ist für alle Personen bis auf die 4m entfernte ziemlich genau. Die Entfernungsschätzung des Gesichtsdetektors ist für alle Personen bis auf 20cm genau. Die Entfernungsschätzung der HOG-Basierten Detektoren ist fast immer zu hoch und bei 3-5m konzentriert.

5.6.3 Bewertung

Die Detektionen, welche von der schwenkenden Kamera abhängen haben im Vergleich zum Laser ein mit 48° auf weniger als die Hälfte eingeschränktes Sichtfeld. Dennoch treten die kamerabasierten Detektionen in dem dicht besetzen Szenario mit hoher Häufigkeit auf.
Die Entfernungsschätzung der Laserdaten funktioniert bis zu 3,5m recht ordentlich. Darüber hinaus sollte man sie nicht verwenden, eine Einschränkung unter 4,0m hilft, die Fehlerrate zu reduzieren. Die Entfernungsschätzung des Gesichtsdetektors ist für alle Personen recht präzise.
Der Ganzkörperdetektor kann lediglich für Person C und bei günstiger Kamerausrichtung für Person B sinnvolle Werte liefern, da nur hier der Fußboden in das Sichtfeld der Kamera fällt. Hier ist die Entfernungsschätzung auch auffallend gut, allerdings wird Person C nicht fokussiert, da der Beinpaardetektor sie nur sehr selten erfasst. Darüber hinaus ist die Entfernungsschätzung der HOG-Basierten Detektoren extrem ungenau und führt zu weiten Entfernungen. Dies geht auf die schon beschriebenen Fehldetektionen zurück, siehe auch Abbildung 50. Wie aus Abbildung 62 ersichtlich wird, führt in diesem Fall eine konjunktive Verknüpfung mit den Beinpaardetektionen zur Unterdrückung der Fehldetektionen, da schlicht keine passenden Beinpaardetektionen auftreten.

5.7 Evaluierung des Clusterings

Für die Evaluierung des Clusteralgorithmus wurde eine Schnittstelle implementiert, die es ermöglicht, reale Detektionen als Objekte zu serialisieren und zu deserialisieren. So ist es möglich anhand der selben Eingabewerte verschiedene Konfigurationen zu vergleichen. Ziele der Evaluierung sind gute Toleranzbereiche der Detektoren zu ermitteln, die Berechnung des Konfidenzwertes zu überprüfen sowie die Möglichkeit zu un-

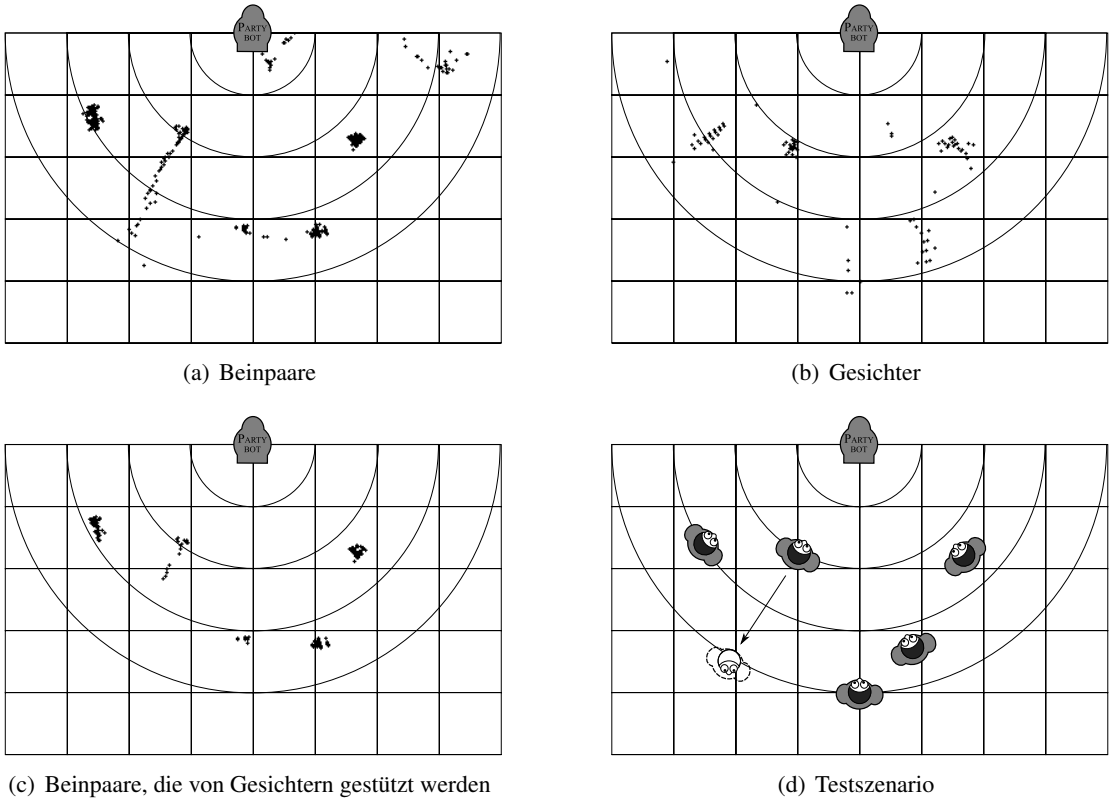
Detektor	Breite [m]	Tiefe [m]
Beinpaare	0,2	0,2
Gesichter	0,3	0,5
Oberkörper	0,5	2,0
Ganzkörper	0,5	2,0

Tabelle 19: Ermittelter Toleranzbereich der Detektoren

tersuchen, Detektoren bei der Entfernungsbestimmung und Clusterbildung zu priorisieren. Für die Bestimmung der Toleranzbereiche wurden manuell Detektionen einer einzelnen Person verglichen und die Abweichung von der tatsächlichen Position betrachtet. Der Beinpaardetektor liefert dabei sehr gute Abschätzungen in der Entfernung und im Drehwinkel. Er benötigt daher einen kleineren Toleranzbereich. Die Positionsschätzung der Gesichtsdetektion basiert, wie in Abschnitt 5.3 beschrieben, auf einer durchschnittlichen Gesichtshöhe und ist daher bei realen Gesichtern etwas ungenauer. Sie wird deshalb großzügiger betrachtet. Der Ganzkörper- und Oberkörperdetektor ist in der Entfernungsmessung sehr ungenau. Hier wird deshalb ein großer Toleranzwert eingestellt. Die ermittelten Breiten- und Tiefenwerte der Toleranzbereiche sind in Tabelle 19 dargestellt.
Zur Überprüfung der Berechnung des Konfidenzwertes wurden verschiedene Kombinationen von Detektionen verglichen. Während der Integrationstests hat sich gezeigt, dass wir gute Ergebnisse erzielen, wenn wir den Detektoren bei der Berechnung die gleiche Gewichtung geben. Cluster mit nur einer Detektion werden um 40% abgewertet, solche der Größe 2 bleiben unbeinflusst und Cluster mit mehr als 2 verschiedener Detektion einer Person erreichen einen Konfidenzwert von 1,0. Durch den Einfluss der Clustergröße können auch Detektionen mit niedrigem Konfidenzwert den kombinierten Wert eines Clusters positiv beeinflussen. Ein Cluster bestehend aus einer Beinpaardetektion mit einer niedrigen Konfidenz von 0,2 und einer Gesichtsdetektion bekommen so eine ähnliche Gewichtung wie ein Cluster, welches nur aus einer Beinpaardetektion mit 0,99 besteht.
Es besteht die Möglichkeit, einen Detektionstyp bei der Bestimmung der Entfernung eines Clusters zu bevorzugen. Es hat sich gezeigt, dass es sinnvoll ist, diese Konfiguration zu nutzen und die Abschätzung des Beinpaardetektors zu priorisieren. Dieser lieferte dazu unter den Detektoren die besten Ergebnisse. Zusätzlich führt die vage Abschätzung des Ganzkörper- und Oberkörperdetektors zu schlechteren Positionannahmen gegenüber der wirklichen Position der Person. Beim Drehwinkels hat sich gezeigt, dass die Berechnung des Mittelwertes gute Ergebnisse liefert.
Die Berechnung der Detektionscluster kann auf einen Detektionstyp gestützt werden. Dabei werden allerdings nur Cluster gebildet, welche eine Detektion dieses Typs beinhalten. Detektionen, die nicht mit einem solchen kombiniert werden können, werden verworfen. Dieses erschien nur bei der Beinpaardetektion sinnvoll, wenn es notwendig ist, dass jeder Cluster eine gute Abschätzung der Entfernung liefern soll.

5.7.1 Bewertung

Bei Testläufen hat sich gezeigt, dass wir mit den Werten in Tabelle 19 gute Ergebnisse erhalten. Auch die Berechnung des Konfidenzwertes stellte sich als sinnvoll heraus. Personen, die von mehreren Detektoren erkannt wurden, können so bevorzugt behandelt werden. Da wir die Positionsangaben eines Clusters für



(a) Beinpaare (b) Gesichter

(c) Beinpaare, die von Gesichtern gestützt werden (d) Testszenario

Abbildung 63: Detektionskombination aufgrund räumlicher Nähe. Punkte bezeichnen Detektionen in einer Fläche von 8x5m vor dem PARTYBOT.

die kartenbasierte Navigation nutzen, ist es erforderlich, dass die Entfernungsangaben sehr genau sind. Der Clusteralgorithmus liefert nur brauchbare Ergebnisse, wenn wir den Beinpaardetektor bei der Entfernungsbestimmung bevorzugen. Weiter sind solche Cluster, die nur bildbasierte Detektionen beinhalten, nicht zur kartenbasierten Navigation brauchbar. Bei diesen wird weiterhin die Position über den Durchschnitt berechnet. Deshalb hat es sich als sinnvoll erwiesen, auch die Ermittlung der Cluster auf den Beindetektor basieren zu lassen. Mit dieser Konfiguration werden die Beinpaardetektionen über die bildbasierten Detektoren verifiziert. In Abbildung 63 wird an einem Beispiel gezeigt, wie diese Verifikation die Detektionen einschränkt. Bei anderen Anwendungsfällen kann die Strategie gewechselt werden. Wenn zum Beispiel nur der ungefähre Richtungswinkel einer Person benötigt wird, sollte auf die letzte Einstellung verzichtet werden.

5.8 Versuche mit der dynamischen Wegplanung

In diesem Kapitel wird die dynamische Wegplanung getestet und in zwei Szenarien unter zwei Parametereinstellungen mit ihr experimentiert. Eine ausführlichere Evaluierung der dynamischen Wegplanung wäre gern gesehen, aber aufgrunddessen, dass wir nur einen Ort haben, in dem wir testen konnten (Finca des IRF in Abbildung 64 rot markiert), war diese leider nicht möglich. Als dynamische Hindernisse werden die beiden Türen (A und B) benutzt, die in der Karte als offene Durchgänge zu sehen sind.
Die zwei Parameter sollen hier noch einmal kurz erklärt werden:

- *functionality 1:* Unter dieser Einstellung werden Winkelfehler und die falsche X-Y-Position behoben. Jedoch reagiert der PARTYBOT empfindlich auf dynamische Umgebungen und starke Drehung, wodurch es noch immer schnell zu einer falschen Lokalisierung kommt.

- *functionality 2:* Hierbei werden nur Daten erfasst, solange sich der Roboter fast geradlinig bewegt. Dies heißt besonders, dass der PARTYBOT seine Umgebung nicht betrachtet, wenn er stehen bleibt. Dennoch ist die Lokalisierung von ausreichender Güte um Fehler zu vermeiden.

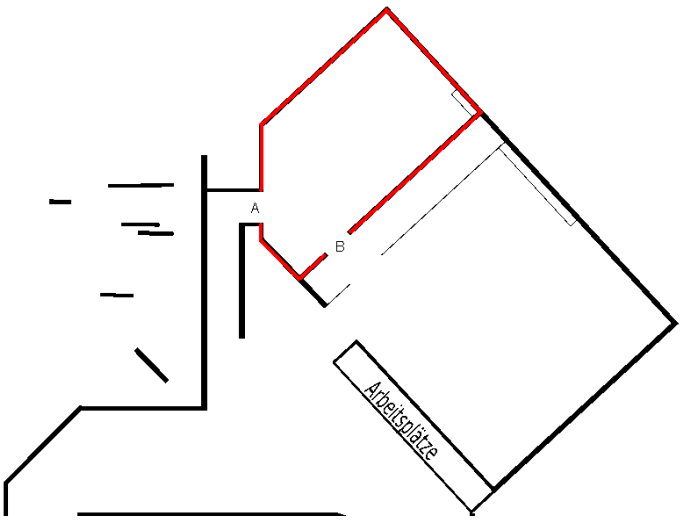
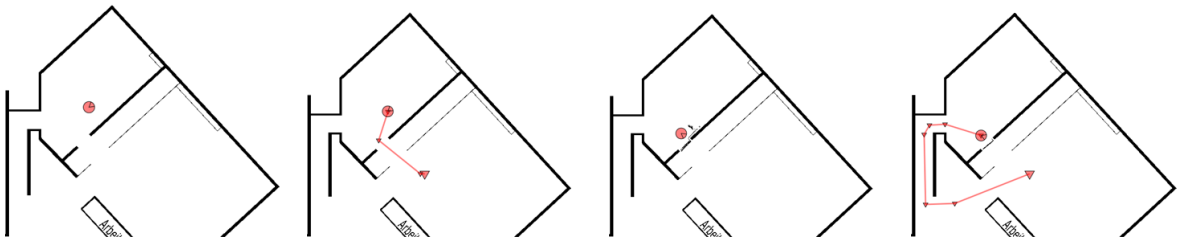


Abbildung 64: Karte der Umgebung am IRF, in welcher der Roboter eingesetzt wird

5.8.1 Szenario 1

Im ersten Szenario ist die Tür (Tür B) geschlossen, die auf dem geplanten Weg liegt. Der erste Test verläuft unter der Einstellung „funtionality 1“ (siehe C.1).
Der PARTYBOT wurde in seiner Ausgangslage so ausgerichtet, dass er keine von den beiden Türen mit seinem Laser erfasst (siehe Abbildung 65(a)). Nun soll der Roboter sein erstes Ziel anfahren und bekommt die Koordinaten des Zielpunktes. Diese liegen im Nebenraum, so dass er auf direktem Weg durch die Tür B plant (siehe Abbildung 65(b)). Während er sich auf die Tür zubewegt, erkennt der PARTYBOT diese als ein Hindernis und zeichnet es auf seinem Weg immer stärker in die dynamische Karte ein. Wenn der Roboter vor dem Hindernis steht, hat er dies zwar eingezeichnet, versucht aber erstmal weiterhin um das Hindernis herum zu planen. Dadurch entsteht eine Art „Wackeln“ des Roboters, da er erst versucht auf der einen Seite vorbeizukommen und bei erkennen des Hindernisses und gleiches auf der anderen Seite (siehe Abbildung 65(c)). Dieses „Wackeln“ wird abgefangen und so darauf reagiert, dass der Roboter den Weg neuplant, so dass keine weitere Hindernisse auf seinem Weg liegen (siehe Abbildung 65(d)).
Im Test 1.2 wird die Einstellung „functionality 2“ vorgenommen. Auch hiermit plante der Roboter seinen Weg neu, nachdem er an dem Hindernis angekommen ist. Diesmal mit dem kleinen Unterschied, dass der PARTYBOT das Hindernis schon beim Drehen in die Richtung des neugeplanten Weges vergessen hat. In diesem Szenario haben die zwei Einstellungen kaum Unterschiede in dem Ergebnis gezeigt.



(a) Startpunkt (b) geplanter Weg (c) Hinderniserkennung (d) neugeplanter Weg

Abbildung 65: Evaluierungsszenario mit einer geschlossenen Tür als dynamisches Hindernis von dem Startpunkt bis zur Wegneuplanung

5.8.2 Szenario 2

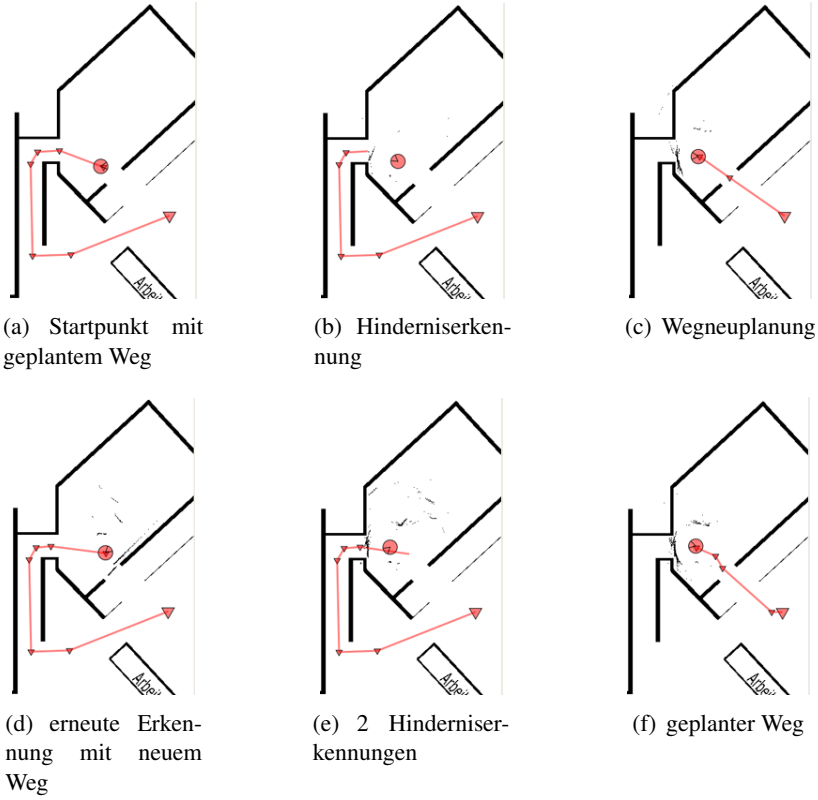
In dem zweiten Szenario sind beide Türen geschlossen und der Roboter versucht, einen Weg aus dem Raum heraus zu planen. In diesem Szenario ist es nicht das primäre Ziel, zu überprüfen, ob der Roboter seinen Weg neuplanen kann, sondern wie der PARTYBOT darauf reagiert aus einem Raum zu planen, bei dem es keinen Weg hinaus gibt. Test 2.1 verläuft wiederum unter dem eingestellten Parameter „functionality 1“.

Die Ausgangssituation hierbei, dass der Roboter direkt vor der Tür B steht, die er unter dieser Einstellung mittels Laserdaten direkt erfasst und als Hindernis erkennt. Der PARTYBOT versucht abermals um das Hindernis herum zu planen ohne allzu weit von seinem ursprünglichen Weg abzuweichen. Es kommt wiederum zu dem „Wackeln“ des Roboters und führt zu dem Neuplanen durch die Tür A. Vor der Tür A kommt es zu dem gleichen Verhalten wie zuvor und der PARTYBOT plant wieder durch die Tür B. Dieses Verhalten des Roboters verläuft solange bis er beide Türen gleichzeitig als Hindernis erkennt oder eine der beide Türen geöffnet wird. Unter der Einstellung „functionality 1“ hat der PARTYBOT beide Türen erfasst und seine Route zu dem Ziel abgebrochen.

In Test 2.2 hat der Roboter wieder die gleiche Ausgangssituation wie in Test 2.1 und die Parametereinstellung „functionality 2“. Wie man in Abbildung 66(a) erkennt, hat der PARTYBOT die erste Tür erfasst, als Hindernis erkannt und seinen Weg neugeplant. Im Gegensatz zu der Einstellung „functionality 1“ wird das Altern auch beim Drehen vollzogen und sogar während der Roboter nur auf der Stelle steht. Allerdings werden keine neuen Daten erfasst und auf die Karte gezeichnet, wodurch es dazu kommt, dass der Durchgang an Tür B in der Abbildung wieder offen ist. Aus Abbildung 66(c) geht hervor, dass er auch die zweite Tür erfasst und neuplant. Jedoch hat er die erste Tür schon wieder „vergessen“ und plant daher den neuen Weg durch sie hindurch.

Dieses Verhalten, Erkennen des Hindernisses und Neuplanen geht unter „functionality 2“ (siehe C.1) schneller als unter dem in dem Szenario 1 eingestellten Parameter, da die Hindernisse verschwinden sobald sich der PARTYBOT dreht. Unter der Einstellung „functionality 1“ musste der Roboter sich erst bewegen, damit die eingezeichneten Pixel altern. Der Roboter hat hier somit das alte Hindernis vergessen sobald er anfängt das neue einzuzeichnen und kann eher einen neuen Weg planen und befahren.

In diesem Szenario kam es aufgrund der Einstellung „functionality 2“ nicht dazu, dass der PARTYBOT beiden Türen gleichzeitig als Hindernis erfasst hat, um somit seine Planung abzubrechen. Der Roboter fährt somit zwischen beide Türen hin und her bis er manuell angehalten wird oder sich eine der beiden Türen öffnet und er herausfahren kann.



(a) Startpunkt mit geplantem Weg

(b) Hinderniserkennung

(c) Wegneuplanung

(d) erneute Erkennung mit neuem Weg

(e) 2 Hinderniserkennungen

(f) geplanter Weg

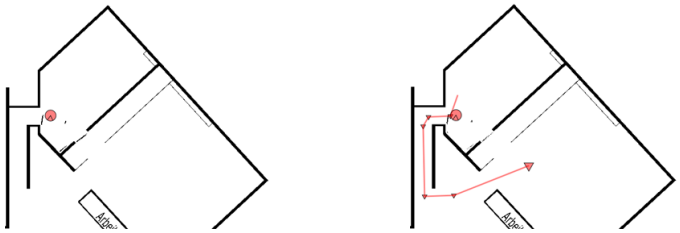
Abbildung 66: Evaluierungsszenario mit zwei geschlossenen Türen von dem Startpunkt bis zur Wegneuplanung unter der Einstellung „functionality 2“

5.8.3 Schwierigkeiten bei der Evaluierung

Eine Schwierigkeit, die im Verlauf der Evaluierung aufgetreten ist, ist die falsche Lokalisierung. Dies bedeutet, dass der Roboter sich selbst an einer Position sieht, an der er nicht steht. Seine korrekte Position weicht von 10cm bis zu 50cm davon ab. Aus Abbildung 67 geht hervor, dass der Roboter an einer Position steht, von der aus er beide Türen sieht, da beide auf der Karte eingezeichnet werden. Die Lokalisierung zeigt aber, dass der Roboter direkt vor der Tür A steht. Die Differenz zwischen Position der Lokalisierung und eigentlicher Position ist ungefähr 35cm. Da der Roboter die Tür A nicht als Hindernis erkannt hat, plant der PARTYBOT einen Weg zu seinem Ziel, dass er gar nicht erreichen kann (siehe Abbildung 66). Die falsche Lokalisierung tritt hauptsächlich unter Verwendung der „functionality 1“ auf, da der Roboter hier bei Drehungen die Daten miterfasst und sie falsch einzeichnet. Unter „functionality 2“ werden keine Daten bei Drehungen erfasst, wodurch diese Variante stabiler ist.

5.8.4 Bewertung

Die unter Punkt 3.2.1 beschriebene stabile erste Variante (functionality 2) des Treibers liefert zufriedenstellende Ergebnisse. Sie eignet sich auch für den Einsatz in einem Party-Szenario. Der Nachteil dieser Variante ist jedoch, dass sie nur Werte betrachtet, wenn der Roboter sich fast geradlinig fortbewegt. Deshalb kann er eine dynamische Umgebung nicht mehr beobachten, wenn er stehen bleibt. Die Lokalisationsfehler werden hier jedoch ausgeglichen, da die meisten Lokalisationen von ausreichender Güte sind. Dadurch treten Fehler



(a) Startpunkt mit Blick auf beide Hindernisse

(b) Pfadplanung trotz gesehener Hindernisse

Abbildung 67: Evaluierungsszenario mit zwei geschlossenen Türen und falscher Lokalisierung

wie in den Abbildungen 19(a) und 19(b) nicht auf.

Die zweite Variante (functionality 1) mit den Erweiterungen zur Behebung der Winkelfehler und der falschen X-Y-Position hat hingegen einige Schwächen. Sie reagiert empfindlich auf dynamische Umgebungen und auf starke Drehungen des Roboters. Sie ist nur begrenzt einsetzbar, da trotz der erzielten Verbesserungen immer noch falsche Lokalisierungen auftreten und eine Fahrplanung unmöglich machen können. Aufgrund der gerade genannten Gründe sehen wir von einer Verwendung dieser Variante ab.

5.9 Systemtests

Schließlich wurde getestet, ob der PARTYBOT gemäß Erwartung Kontakt mit einer alleinstehenden Person aus seiner Umgebung aufnimmt. Falls nur Gruppen von Personen zu finden sind, muss sich der Roboter die geringst entfernte Person aussuchen.

Um das Gesamtsystem zu testen haben wir uns verschiedene Testszenarien ausgedacht, die in Abbildung 68 - 70 veranschaulicht werden. Ein Kästchen aus der Abbildung ist je ein Quadratmeter. Der PARTYBOT wird über eine GUI gesteuert (siehe Anhang B). Wenn das System gestartet wird, ist der PARTYBOT in den Zustand „Idle“, also sucht sich eine Person im Raum aus. Wird diese gefunden, wird deren Position auf der Navigationskarte als Ziel gesetzt und der Roboter gelangt in den Zustand „Fahre zur Person“. Ein Weg zur Person wird mit Hilfe unseres Navigationsalgorithmus geplant (siehe Abschnitt 3). Während der Fahrt wird mehrfach überprüft, ob sich die Person wegbewegt hat. Ist dies der Fall, gelangt der Roboter wieder in den Zustand „Idle“. Ist die Person noch da und der Roboter erreicht sein Ziel, wird in den Zustand „Kontakt aufnehmen“ gewechselt. Dabei testen wir also die Zusammenarbeit der Personendetektion, der Navigation in dynamischen Umgebungen und der Steuerarchitektur.

5.9.1 Szenario 1: Finden und Ansprechen einer Person aus einer Gruppe

Im ersten Testszenario gab es zwei Gruppen von je drei Personen (siehe Abbildung 68). Das bedeutet also, dass der PARTYBOT keine alleinstehende Person finden konnte. Er hat sich stets eine ihm nahestehende Person aus einer der Gruppen ausgesucht. Bei sechs Wiederholungen wurden zweimal Person D und viermal Person C vom PARTYBOT angesprochen.

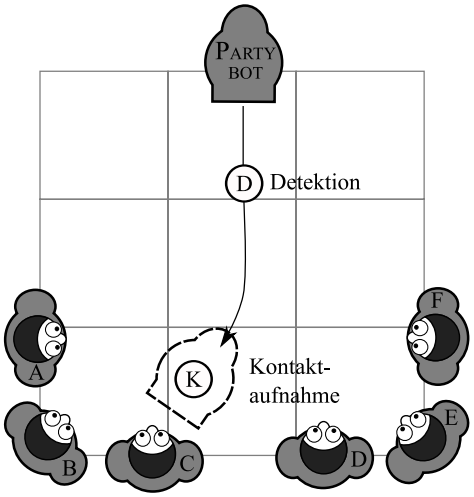
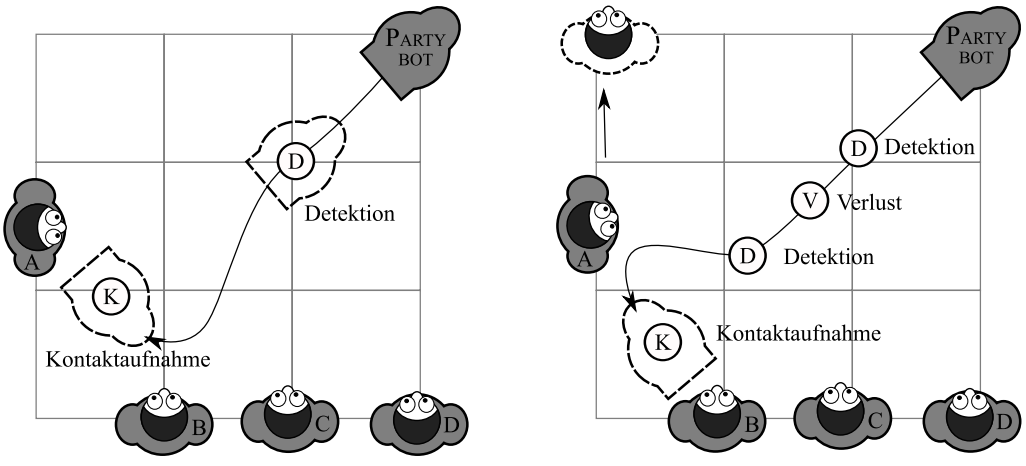


Abbildung 68: Szenario 1: Finden und Ansprechen einer Person aus einer Gruppe



(a) Szenario 2a (b) Szenario 2b

Abbildung 69: Szenario 2: Finden und Ansprechen einer alleinstehenden Person

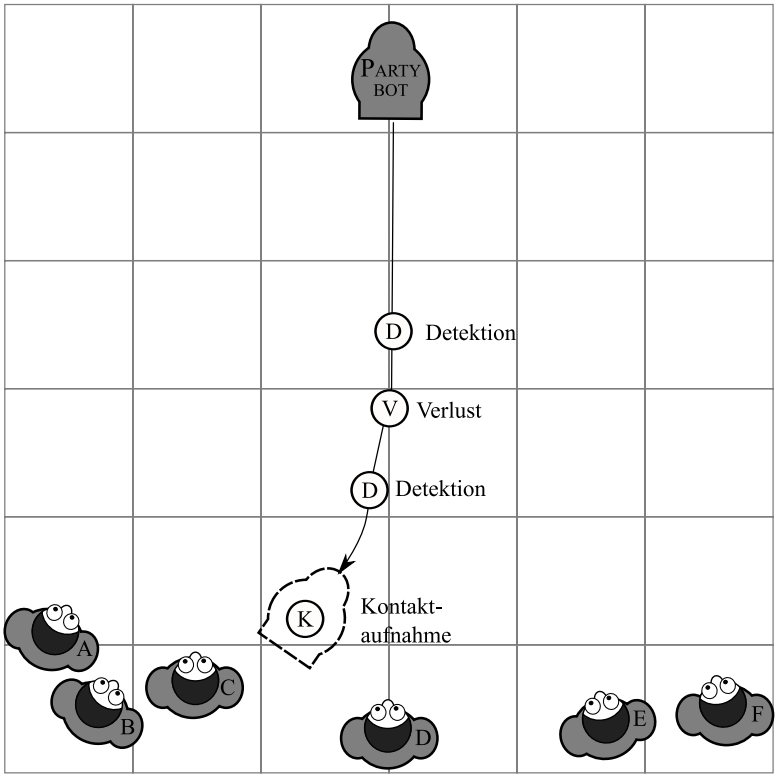


Abbildung 70: Szenario 3: Finden einer entfernten Person

5.9.2 Szenario 2: Finden und Ansprechen einer alleinstehenden Person

Im Falle der alleinstehenden Person musste noch berücksichtigt werden, dass sich die Person wegbewegen konnte, während der Roboter noch auf dem Weg zu ihr war. Dies sollte vom Roboter „bemerkt“ werden, und eine neue Personensuche gestartet werden.

Szenarion 2a: Einzelne Person interessiert

Falls die Person noch da war, gab es zwei Gruppen von je einer und drei Personen (siehe Abbildung 69(a)). Der PARTYBOT suchte sich stets die alleinstehende Person und kontaktierte diese.

Szenarion 2b: Einzelne Person läuft weg

Im Szenario, in dem die Person nicht mehr auf ihrer Position war, gab es auch zwei Gruppen von je einer und drei Personen (siehe Abbildung 69(b)), aber die alleinstehende Person A lief weg, sobald sich der PARTYBOT ihr näherte. Der PARTYBOT verlor also A und gelang wieder in den „Idle“ Zustand, wobei er sich wie erwartet, stets die nächste Person in der Gruppe suchte und diese kontaktierte.

5.9.3 Szenario 3: Finden und Ansprechen einer entfernten Person

Mit dem letzten Szenario (siehe Abbildung 70) wollten wir testen, wie sich der PARTYBOT verhält, wenn alle Personen weit von ihm entfernt stehen. Dafür gab es drei Gruppen von je einer, zwei und drei Personen

in 6m Entfernung, in der der Beindetektor keine Beine erkennen kann.
Der PARTYBOT suchte zunächst stets den Raum ab, entdeckte bei Annäherung die Personen. Auf dem Weg verlor er mehrfach die detektierte Person und suchte sich eine neue, bis er schließlich eine ansprach. Dieses Szenario haben wir viermal probiert, wobei in zwei Fällen auf dem Weg die Detektion verloren und wiedergefunden wurde. Der Roboter kontaktierte einmal Person D und dreimal Person C.

5.9.4 Bewertung

Die integrierte Test aller Funktionalitäten liefert zufriedenstellende Ergebnisse. Personen werden schnell detektiert wobei alleinstehende Personen bevorzugt werden und wenn sich eine Person wegbewegt, wird das vom System erkannt und der Roboter sucht sich eine neue Kontaktperson aus. Bei Personen, die weit entfernt vom Roboter stehen, können Probleme aufgrund der Lokalisierung auftreten, denn wenn sich der Roboter auf dem Weg zur Person falsch lokalisiert, muss die Person wieder detektiert werden.

6 Fazit

Mit dem PARTYBOT wurde ein mobiler und autonomer Roboter geschaffen, dessen Einsatzgebiet Veranstaltungen in geschlossenen Räumlichkeiten mit vielen anwesenden Menschen, wie zum Beispiel Empfänge, Partys und Informationsveranstaltungen, sind. Dort erkennt er selbsttätig Personen und nähert sich ihnen, um Kontakt aufzunehmen. Dabei bewegt er sich frei zwischen den Gästen und sucht nach Personen, mit denen er interagieren kann, und versucht abhängig von den Personen und Hindernissen in seiner Umgebung einen Weg dorthin zu planen. Bei der Kontaktaufnahme priorisiert er einzeln stehende Personen, die durch Blickkontakt ihr Interesse andeuten. Diese können über einen Touchscreen mit dem PARTYBOT kommunizieren und haben somit Zugriff auf verschiedene Dienste. Der Roboter kann auf Wunsch Personen folgen, auch haben die Gäste die Möglichkeit, sich vom PARTYBOT fotografieren zu lassen und das Foto per E-Mail zu erhalten. Nachdem die kontaktierte Person keine weiteren Dienste mehr beansprucht, kehrt der PARTYBOT in seinen Anfangszustand zurück und versucht eine neue Person zur Kontaktaufnahme zu finden. Das Verhalten wird abgerundet durch seine Fähigkeit Gefahren zu erkennen und ggf. Angst zu zeigen. Laute Geräusche oder schnelles Herantreten von Personen kann ein solches Angstverhalten auslösen.

Das Verhalten des Roboters wurde durch eine zustandsbasierte Steuerarchitektur auf Basis von OSGi realisiert. Abhängig von den einzelnen Zuständen werden entsprechende Komponenten initialisiert bzw. abgeschaltet. Als Beispiel muss bei einem Wechsel in den Follow-Me-Modus der Beinpaarerkenner abgeschaltet werden, damit der Laser uneingeschränkt zur Verfügung steht. Die Zustandswechsel werden über Ereignisse gesteuert. Insgesamt hat sich diese Designentscheidung als geeignet erwiesen, da sich alle Funktionalitäten im Hinblick auf die Zielsetzung problemlos umsetzen ließen.

Die Personenerkennung wird über vier verschiedene Detektoren realisiert. Der Beinpaarerkenner erhält seine Daten aus den Laserwerten, wohingegen der Ober-, Ganzkörper- und Gesichtsdetektor videobasiert arbeiten. Zur Gesichtsdetektion wird der Viola & Jones Algorithmus verwendet, der Ober- und Ganzkörperdetektor basieren auf HOG-Deskriptoren. Die Ergebnisse der vier Detektoren werden zusammengefasst und zu einer Personenhypothese kombiniert. Die verschiedenen Reichweiten der Detektoren müssen hier berücksichtigt werden. Der Beinpaarerkenner erkennt Personen im Nahbereich, da bei zunehmender Entfernung die detektierten Elemente pro Winkel abnehmen, wohingegen die HOG-basierten Ober- und Ganzkörperdetektoren nur bei weiter entfernten Personen funktionieren, da die Personen sonst nicht vollständig im Bild sind. Auch die ermittelten Entfernungen der einzelnen Detektoren sind unterschiedlich zu bewerten. Der Beinpaarerkenner liefert aufgrund der Abstandsmessung per Laser sehr genaue Werte. Der Gesichtserkenner ermittelt trotz einer Entfernungsschätzung anhand der Größe der detektierten Gesichter im Nahbereich relativ gute Werte. Somit lässt sich aus den Ergebnissen des Beinpaar- und Gesichtsdetektors eine recht gute Positionsschätzung für die Personenhypothesen berechnen. Die HOG-basierten Ober- und Ganzkörperdetektoren liefern eine gute Winkelschätzung. Sie können daher neben der Erhöhung der Konfidenz im Nahbereich zum Navigieren zu weiter entfernten Personen verwendet werden.

Die Navigation des PARTYBOTS basiert auf der Weiterentwicklung des Wavefrontalgorithmus um dynamische Hindernisse. Da aufgrund des Einsatzszenarios mit vielen Personen die Umgebung sich permanent ändert, musste der statische Ansatz erweitert werden, um auf temporär versperrte Routen zu reagieren. Als ungünstig bei diesem Verfahren hat sich erwiesen, dass die Wegplanung während der Fahrt nicht dynamisch angepasst werden kann. Daher kann der PARTYBOT nur bedingt auf sich bewegende Ziele reagieren, und

muss bei Überprüfung der Personenhypothesen anhalten. Des Weiteren ist die vom Player-Framework zur Verfügung gestellte Lokalisierung störanfällig. Der Grund dafür ist, dass bei hohem Personenaufkommen die Geometrie des Raumes nur bedingt mit dem Kartenmaterial verglichen werden kann. Da die Lokalisierung benötigt wird, um das Erreichen des Zielpunktes zu erkennen, kann dies zur Abweichung zum eigentlich erwünschten Zielpunkt führen. Eine eigene Implementierung kam ressourcenbedingt nicht in Frage.
Schon die erste Demonstration auf dem Schülertag zeigte durch ihr positives Feedback, dass für ein solches Konzept Interesse besteht und der PARTYBOT seinen Namen somit zurecht trägt. Mit dem PARTYBOT ist ein artifizieller Partygast entstanden, der durch seine Fähigkeit auf Personen zu reagieren, sie erkennbar anzusehen und sein Vorhaben lautstark kundzutun einen sympathischen Eindruck auf die Menschen in seiner Umgebung macht.

Literatur

[1] AL-JUMAILY, A. ; LEUNG, C. : Wavefront Propagation and Fuzzy Based Autonomous Navigation. In: *International Journal of Advanced Robotic Systems* 2 (2005), Nr. 2, S. 93–102

[2] ARKIN, R. C.: *Behavior-based Robots*. MIT Press, 1998

[3] ARRAS, K. O. ; MOZOS, O. M. ; BURGARD, W. : Using Boosted Features for the Detection of People in 2D Range Data. In: *Proceedings of the IEEE International Conference on Robotics and Automation, Rom, Italy*, 2007, S. 3402–3407

[4] BAN, S.-W. ; LEE, M. : Biologicallly motivated Visual Selective Attention for Face Localization. In: PALETTA, L. (Hrsg.) ; TSOTSOS, J. K. (Hrsg.) ; ROME, E. (Hrsg.) ; HUMPHREYS, G. W. (Hrsg.): *Attention and performance in computational vision. EC Vision, Prag 2004*, LNCS 3368, Springer, 2005, S. 196–206

[5] BELLOTTO, N. ; HUOSHENG, H. : Multisensor-Based Human Detection and Tracking for Mobile Service Robots. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 39 (2009), Nr. 1, S. 167–181

[6] BISHOP, C. M.: *Neural Networks for Pattern Recognition*. Oxford University Press, 1996

[7] BOTHE, H.-H. : *Neuro-Fuzzy Methoden*. Springer, 1998

[8] BROOKS, R. A.: Intelligence Without Reason. In: MYOPOULOS, J. (Hrsg.) ; REITER, R. (Hrsg.): *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*. Sydney, Australia : Morgan Kaufmann publishers Inc., 1991, S. 569–595

[9] BURGES, C. J. C.: A tutorial on support vector machines for pattern recognition. In: *Data Mining and Knowledge Discovery* 2 (1998), S. 121–167

[10] BYERS, Z. ; DIXON, M. ; GOODIER, K. ; GRIMM, C. M. ; SMART, W. D.: Say cheese! Experiences with a robot photographer. In: *AI Mag.* 25 (2004), Nr. 3, S. 37–46. – ISSN 0738–4602

[11] COMANICIU, D. ; MEER, P. : Mean Shift: A Robust Approach Toward Feature Space Analysis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), S. 603–619

[12] DALAI, N. ; TRIGGS, B. : Histograms of Oriented Gradients for Human Detection. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2005), S. 886–893

[13] DALAL, N. : *Finding people in images and videos*, Institut National Polytechnique de Grenoble, Diss., 2006

[14] DUDA, R. O. ; HART, P. E. ; STORK, D. G.: *Pattern Classification*. John Wiley & Sons, New York, 2001

[15] EVERINGHAM, M. ; ZISSERMAN, A. ; WILLIAMS, C. K. I. ; GOOL, L. V.: *The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results*. `http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf`, 2006

[16] FEYRER, S. : *Detektion, Lokalisierung und Verfolgung von Personen mit einem mobilen Serviceroboter*, Eberhard-Karls-Universität Tübingen, Diss., 2000

[17] FRINTROP, S. ; JENSFELT, P. ; CHRISTENSEN, H. : Simultaneous Robot Localization and Mapping Based on a Visual Attention System. In: *Attention in Cognitive Systems, LNAI 4840*, Springer, 2007, S. 417–430

[18] GRAF, B. ; BAUM, W. ; TRAUB, A. ; SCHRAFT, R. : Konzeption dreier Roboter zur Unterhaltung der Besucher eines Museums. In: *Tagung Robotik 2000 Berlin, Germany*, 2000, S. 529–536

[19] GROSS, H.-M. ; BOEHME, H.-J. ; SCHROETER, C. ; MUELLER, S. ; KOENIG, A. ; MARTIN, C. ; MERTEN, M. ; BLEY, A. : ShopBot: Progress in Developing an Interactive Mobile Shopping Assistant for Everyday Use. In: *Proceedings. 2008 IEEE Int. Conf. on Systems, Man and Cybernetics (SMC 2008) Singapore*, 2008, S. 3471–3478

[20] HANS, M. ; GRAF, B. : *Robotic Home Assistant Care-O-bot II*. Bd. 14. Springer, 2004

[21] HARITAOGLU, I. ; HARWOOD, D. ; DAVIS, L. : W4s: A real time system for detecting and tracking people in 2.5D. In: *European Conference on Computer Vision, Freiburg*, 1998, S. 877–892

[22] HERTZBERG, J. ; KIRCHNER, F. : Landmark-based autonomous navigation in sewerage pipes. In: *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT-96) Kaiserslautern Germany*, 1996, S. 68–73

[23] ITTI, L. ; KOCH, C. ; NIEBUR, E. : A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Nr. 11, S. 1254–1259

[24] JABRI, S. ; DURIC, Z. ; WECHSLER, H. ; ROSENFELD, A. : Detection and location of people in video images using adaptivefusion of color and edge information. In: *15th International Conference on Pattern Recognition, Barcelona, Spain* Bd. 4, 2000, S. 627–630

[25] JAIN, A. K. ; DUIN, R. P. W. ; MAO, J. : Statistical Pattern Recognition: A Review. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 1, S. 4–37

[26] JOHNSON, D. S.: A theoretician's guide to the experimental analysis of algorithms. In: *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, American Mathematical Society, 2002, S. 215–250

[27] KATZENMEIER, H. W.: *Reinigungsroboter selbstgebaut*. Elektor-Verlag; Auflage: 1 (19. Juni 2006), Deutschland, 2006

[28] LANG, S. ; KLEINEHAGENBROCK, M. ; FRITSCH, J. ; FINK, G. A. ; SAGERER, G. : Detection of Communication Partners from a Mobile Robot. In: *4th Workshop on Dynamic Perception, Bochum*, 2002, S. 183–188

[29] LANG, S. : *Multimodale Aufmerksamkeitssteuerung für einen mobilen Roboter*, Universität Bielefeld, Technische Fakultät, Diss., 2005

[30] LAVALLE, S. M.: *Planning Algorithms*. Cambridge University Press, 2006

[31] LEE, C. Y.: An algorithm for path connections and its applications. In: *IRE Transactions on Electronic Computers EC-10* (1961), S. 346–365

[32] LIENHART, R. ; MAYDT, J. : An Extended Set of Haar-like Features for Rapid Object Detection. In: *IEEE International Conference on Image Processing 2002* Bd. 1, 2002, S. 900–903

[33] LUH, G.-C. ; LIU, W.-W. : Potential Field Based Immune Network for Dynamic Motion Planning of Mobile Robots. In: *Strategic Technology, The 1st International Forum on* (2006), Oct., S. 151–155

[34] LUMELSKY, V. ; STEPANOV, A. : Dynamic path planning for a mobile automaton with limited information on the environment. In: *Automatic Control, IEEE Transactions on Automatic Control* 31 (1986), Nr. 11, S. 1058–1063. – ISSN 0018–9286

[35] MARTIN, C. ; SCHAFFERNICHT, E. ; SCHEIDIG, A. ; GROSS, H.-M. : Sensor Fusion Using a Probabilistic Aggregation Scheme for People Detection and People Tracking Robotics and Autonomous Systems. In: *Robotics and Autonomous Systems* 54 (2006), S. 721–728

[36] MATARIC, M. J.: *Learning in behavior-based multi-robot systems: policies, models, and other agents*. Elsevier Science B.V., 2001

[37] MURPHY, R. : *Introduction to AI Robotics*. Cambridge, MA, USA : MIT Press, 2000

[38] NIEMANN, H. : *Klassifikation von Mustern*. Springer, 1983

[39] *OpenCV Wiki*. http://opencv.willowgarage.com/wiki/ besucht am 10.10.2008,

[40] OKUNO, H. G. ; NAKADAI, K. ; KITANO, H. : Social interaction of humanoid robot based on audio-visual tracking. In: *18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2002)*, 2002, S. 140–173

[41] OSADCHY, M. ; CUN, Y. L. ; MILLER, M. L.: Synergistic Face Detection and Pose Estimation with Energy-Based Models. In: *Jorunal of Machine Learning Research* 8 (2007), S. 1197–1215. – ISSN 1533–7928

[42] THE OSGI ALLIACE (Hrsg.): *OSGi Service Platform Core Specification, Release 4, Version 4.1*. The OSGi Alliace, 2007

[43] PAPAGEORGIOU, C. P.: A Trainable System for Object Detection in Images and Video Sequences / Artificial Intelligence Laboratory, Massacuchetts Institute of Technology. 2000 (1685). – Forschungsbericht

[44] PAPAGEORGIOU, C. P. ; POGGIO, T. : A Trainable System for Object Detection. In: *International Journal of Computer Vision* 38 (2000), Nr. 1, S. 15–33

[45] PAULUS, D. W. R. ; HORNEGGER, J. : *Applied Pattern Recognition – A practical Introduction to Image and Speech Processing in C++*. Vieweg, 2001

[46] PIRJANIAN, P. : Behavior Coordination Mechanisms - state-of-the-Art / Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California. 1999 (IRIS-99-375). – Forschungsbericht

[47] PLATT, J. C.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. Version: 1998. http://portal.acm.org/citation.cfm?id=299105. In: SCHÖLKOPF, B. (Hrsg.) ; BURGES, C. (Hrsg.) ; SMOLA, A. (Hrsg.): *Advances in Kernel Methods - Support Vector Learning*. MIT Press. – ISBN 0262194163, 185–208

[48] *Player Project*. Internetseite, 2008. – Erreichbar unter http://playerstage.sourceforge.net, besucht am 11.11.2008.

[49] ROWLEY, H. ; BALUJA, S. ; KANADE, T. : Rotation invariant neural network-based face detection. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, S. 38–44

[50] ROWLEY, H. A. ; BALUJA, S. ; KANADE, T. : Neural Network-Based Face Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Nr. 1, 23–38. http://citeseer.ist.psu.edu/rowley96neural.html

[51] RUESCH, J. ; LOPES, M. ; BERNARDINO, A. ; HORNSTEIN, J. ; SANTOS-VICTOR, J. ; PFEIFER, R. : Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In: *IEEE International Conference on Robotics and Automation, Pasadena, California*, 2008, S. 962–967

[52] SCHNEIDER, J. : *Does Face Detection Technology Really Work?* Adourama Imageing Resource Center, http://www.adorama.com/catalog.tpl?article=052107&op=academy_new, 2007

[53] SCHRAFT, R. ; WEGENER, K. ; SIMONS, F. ; PFEIFFER, K. : *Intelligent Sensor System and Flexible Gripper for Security Robots*. Bd. 13. Springer, 2006

[54] SCHRAFT, R. D. ; HÄGELE, M. ; WEGENER, K. : *Service Roboter - Visionen*. Hanser Fachbuchverlag, 2004

[55] SOLINA, F. ; PEER, P. ; BATAGELJ, B. ; JUVAN, S. : 15 seconds of fame - an interactive, computer-vision based art installation. In: *7th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore*, 2002, S. 198–204

[56] STEELS, L. : *The artificial life roots of artificial intelligence*. Artificial Life Journal, 1(1), 1994

[57] STENZEL, R. : *Steuerungsarchitekturen fur autonome mobile Roboter, Fakultät für Mathematik, Informatik und Naturwissenschaften*, RWTH Aachen, Diss., 2002

[58] STOLLNITZ, E. ; ROSE, T. de ; SALESIN, D. : *Wavelets for Computer Graphics – Theory and Applications*. Morgan Kaufman Publ. Inc., 1996

[59] THRUN, S. : *Learning metric-topological maps for indoor mobile robot navigation*. Computer Science Department and Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1997

[60] TREISMAN, A. ; GELADE, G. : A Feature–Integration Theory of Attention. In: *Cognitive Psychology* 12 (1980), S. 97–136

[61] VIOLA, P. ; JONES, M. : Rapid Object Detection using a Boosted Cascade of Simple Features. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 1, 2001, S. 511–518

[62] XAVIER, J. ; PACHECO, M. ; CASTRO, D. ; RUANO, A. ; NUNES, U. : Fast Line, Arc/Circle and Leg Detection from Laser Scan Data in a Player Driver. In: *IEEE International Conference on Robotics and Automation, Barcelona, Spain*, 2005, S. 3930–3935

[63] YAMAUCHI, B. ; POOK, P. ; GRUBER, A. : Bloodhound: A Semi-Autonomous Battlefield Medical Robot. In: *Proceedings of the 23rd Army Science Conference*. U.S. Army, Orlando, FL, USA, 2002, S. 68–73

[64] YANG, G. ; HUANG, T. S.: Human face detection in a complex background. In: *Pattern Recognition* 27 (1994), Nr. 1, S. 53–63

[65] YANG, M.-H. ; KRIEGMAN, D. J. ; AHUJA, N. : Detecting faces in images: a survey. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), Nr. 1, S. 34–58

[66] ZEKI, S. : *A Vision of the Brain*. Wiley-Blackwell, 1993

[67] ZHANG, J. ; MARSZALEK, M. ; LAZEBNIK, S. ; SCHMID, C. : Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. In: *Internationl Journal on Computer Vision* 73 (2007), Nr. 2, S. 213–238. – ISSN 0920–5691

Abbildungsverzeichnis

1 Konzeptionelle Aufteilung der Themenbereiche . . . 10
2 Der SCITOS G5 mit Kamera . . . 11
3 Entfernungsarbeitsbereich der Detektoren (Sicht von oben) . . . 13
4 Extraktion der Segmente aus den Laserdaten . . . 17
5 Arbeitsweise des Gesichtsdetektors . . . 21
6 Vom Gesichtsdetektor verwendete Merkmale . . . 22
7 Arbeitsweise der HOG-basierten Detektoren . . . 25
8 Berechnung des orientierten Gradienten in einer 3x3 Zelle. . . . 26
9 Gradientenbild einer Testaufnahme . . . 27
10 Durschnittsgradient und HOG . . . 28
11 Abschätzung der Fußbodenhöhe . . . 29
12 Zusammenfassung benachbarter Detektionen . . . 31
13 Toleranzen und zusammenfassbare Detektionen beim Clustering . . . 34
14 Problemstellung: dynamisch blockierter Pfad (rot gestrichelt), alternativer Pfad (grün) . . . 37
15 Ausgangslage des Beispielszenarios . . . 38
16 Funktionsweise des Wavefront-Algorithmus . . . 39
17 Integration des von uns entwickelten Treibers Dynamicmapping in das Player-Framework . . . 40
18 Verschmelzen der Umgebungskarte und der dynamischen Karte mit anschließender Normalisierung . . . 40
19 Beispiele für eine falsch detektierte Position . . . 42
20 Einzelschritte des Beispielszenarios . . . 44
21 Reaktive Navigation: Ausweichen (links) und Verfolgen (rechts) . . . 46
22 Prozessdatenverarbeitung [57] . . . 47
23 Reaktiver Ansatz . . . 48
24 Deliberativer Ansatz . . . 49
25 Hybrider Ansatz . . . 49
26 Klassifikation der Koordinationsverfahren für verhaltensbasierte Steuerarchitekturen [56] . . . 51
27 Schematischer Aufbau der Gesamtarchitektur . . . 53
28 Eventgetriebener Zustandsautomat als zentrale Steuerinstanz . . . 54
29 Aufbau einer serviceorientierten Architektur nach [42] . . . 55
30 OSGi Layermodell nach [42] . . . 56
31 OSGi Lifecycle nach [42] . . . 57
32 Der PARTYBOT auf dem Schülertag . . . 64
33 Entwurfszyklen eines Mustererkennungsystems . . . 64
34 Kreuzvalidierung . . . 66
35 Kreuzvalidierung mit Subvalidierung für MLP . . . 66
36 Beispiel zur Bewertung von Detektionen abhängig von Annotationen . . . 66
37 Evaluierung von HOG Konfigurationen für Ganzkörper . . . 68
38 Evaluierung von HOG Konfigurationen für Oberkörper . . . 68
39 Variation der Netztopologie . . . 70
40 HOG ROC Neuronenschwellwert . . . 71
41 Evaluierung von Suchraumschritten für Ganzkörper . . . 73
42 Evaluierung von Suchraumschritten für Oberkörper . . . 73
43 Test der Abschätzung der Fußbodenhöhe . . . 74
44 Evaluierung von Mean-Shift-Cluster Gaußkernel-Bandbreite (σ) für Ganzkörper . . . 76
45 Evaluierung von Mean-Shift-Cluster Schwellwerten für Ganzkörper . . . 76
46 Evaluierung von Mean-Shift-Cluster Gaußkernel-Bandbreite (σ) für Oberkörper . . . 77

47 Evaluierung von Mean-Shift-Cluster Schwellwerten für Oberkörper . . . 77
48 HOG ROC Clustersupport . . . 78
49 Ergebnisse der HOG Basierten Detektoren nach Optimierung der Parameter . . . 80
50 HOG Fehldetektionen . . . 80
51 Evaluierung von Cluster Schwellwerten für Gesichter . . . 81
52 Evaluierung von minimale Größe der Gesichter . . . 81
53 Evaluierung von verschiedenen Haarkaskaden für Gesichter . . . 82
54 Bild aus dem Szenario „WhiteRoom“ . . . 83
55 Bild aus dem Szenario „Foyer“ . . . 83
56 Evaluierung der Bildbasierten Detektoren im Szenario „WhiteRoom“ . . . 84
57 Evaluierung der Bildbasierten Detektoren im Szenario „Foyer“ . . . 84
58 Evaluierungsszenario 1 . . . 86
59 richtige gegen falsche Detektionen aufgetragen . . . 89
60 detektierte Varianzen und Beinbreiten in den beiden Szenarien . . . 90
61 Detektorevaluierungsszenario . . . 91
62 Detektionsanzahlen je Raumposition im Test mit Kamerabewegung . . . 92
63 Detektionskombination . . . 95
64 IRF-Karte . . . 96
65 Evaluierungsszenario mit einer geschlossenen Tür als dynamisches Hindernis von dem Startpunkt bis zur Wegneuplanung . . . 97
66 Evaluierungsszenario mit zwei geschlossenen Türen von dem Startpunkt bis zur Wegneuplanung unter der Einstellung „functionality 2“ . . . 98
67 Evaluierungsszenario mit zwei geschlossenen Türen und falscher Lokalisierung . . . 99
68 Systemtest Szenario 1 . . . 100
69 Systemtest Szenario 2 . . . 100
70 Systemtest Szenario 3 . . . 101
71 GUI Detektoren . . . 114
72 GUI Hallo-Tab . . . 115
73 Daten vom Bildserver . . . 117
74 FaceDetection . . . 120
75 Player Schema . . . 121
76 Playerframework . . . 121

Tabellenverzeichnis

1 Mögliche Personenentfernungen in Abhängigkeit vom Kamerawinkel . . . 30
2 Eigenschaften der Detektoren . . . 33
3 Vergleich von SVM Kernen und Typen (Ganzkörper; 1:4 xy, N=4250) . . . 70
4 Vergleich der Ergebnisse mit und ohne Fußbodeneinschränkung . . . 72
5 Evaluierung von Clustering-Schwellwerten für Ganzkörper . . . 75
6 Evaluierung von Clustering-Schwellwerten für Oberkörper . . . 75
7 Test der bildbasierten Detektoren im Szenario „WhiteRoom“ . . . 85
8 Test der bildbasierten Detektoren im Szenario „Foyer“ . . . 85
9 verwendete Parameter . . . 86
10 detektierte Objekte . . . 87
11 detektierte Personen . . . 88
12 Mindestbeinbreiten . . . 88
13 Beindetektionen Personen . . . 88

14 Beindetektionen Nichtpersonen 88
15 Precision und Recall des ersten Evaluierungsszenarios im Vergleich 88
16 Anzahl der Detektionen im Test mit Kamerabewegung 92
17 Winkel der Detektionen im Test mit Kamerabewegung 93
18 Entfernung der Detektionen im Test mit Kamerabewegung 93
19 Toleranzbereich der Detektoren 94
20 Wichtige Bildabrufbefehle des Kameraservers 117
21 Wichtige Steuerbefehle des Kameraservers 118

A SCITOS Bedienung

Eingeschaltet wird die Roboterplattform über einen Hauptschalter, welcher durch einen Schlüssel gesichert ist. Des Weiteren verfügt der SCITOS G5 über ein Statusdisplay, mit dessen Hilfe man die grundlegenden Einstellungen und Funktionen abrufen kann. Durch die Menüs des Statusdisplay wird mit Hilfe eines Drehknopfes navigiert.

Start-Menü Nachdem der Roboter eingeschaltet wurde, befindet man sich im Start-Menü. Dort kann ausgewählt werden, ob nur die Roboterplattform oder zusätzlich der PC miteingeschaltet werden soll.

Main-Menü Im Main-Menü stehen drei Untermenüs zur Auswahl. Das Drive-Menü, das Switching-Menü und das EBC-Menü. Außerdem kann hier der Ladezustand der Batterie sowie die Spannungsversorgung des Roboters abgelesen werden.

Drive-Menü Hier können die aktuellen Daten der Odometrie abgerufen und ggf. zurückgesetzt werden.

Switching-Menü Über das Switching-Menü kann der Motor angehalten bzw. nach einem Notstopp durch die Aktivierung des Bumpers wieder freigegeben werden. Der Free-Run-Modus kann aktiviert werden, wenn der Roboter per Hand geschoben werden soll (in diesem Fall werden die Antriebsräder vom Motor entkoppelt). Außerdem kann der Sonarring ein- bzw. ausgeschaltet werden.

EBC-Menü Das EBC-Menü dient zur Steuerung der Stromversorgung der externen Module. Hier kann die Spannung der Busanschlüsse ein- und ausgeschaltet werden.

B GUI

Das gesellige Verhalten des Roboters ist eines der Ziele der Projektgruppe. Der PARTYBOT soll sich zwischen den Personen frei bewegen können, sich ein Interaktionspartner aussuchen um mit ihn Kontakt aufzunehmen. Dabei bietet er dieser Person verschiedene „Dienste“ an, wie zum Beispiel die Möglichkeit ein Foto zu machen und es per Email zu versenden. Um das alles möglich zu machen und dabei auch die Funktionalität des gesamten Systems zu testen, wurde eine grafische Bedienoberfläche erstellt.
Befindet sich der PARTYBOT im Suchmodus („Idle“ Zustand) ist der Detektoren-Tab am Bildschirm zu sehen. Hier werden die Detektionen (Beinpaare, Gesichter, Oberkörper und Ganzkörper) mit farbigen Punkten in die Grafik angezeichnet. Im Textfeld sind dann weitere Informationen über die gefundenen Detektionen zu finden (siehe Abbildung 71).
Erreicht der Roboter die gefundene Person, wechselt die Aufmerksamkeitsarchitektur den Zustand des Robotors zu „Kontakt aufnehmen“ und zeigt die Begrüßungsseite auf dem Touchscreen an (siehe Abbildung 72). Ist die Person aber nicht interessiert, kann sie den Roboter durch Bestätigen des „Geh weg!“ Knopfes einen anderen Kommunikationspartner suchen lassen. Akzeptiert die Person den Roboter als Interaktionspartner, kann sie über das Touchpad, dem Roboter verschiedene Anweisungen, wie z.B. Folge mir, Foto machen, Angst zeigen, geben. Der PARTYBOT kann auch in den Pause Modus gesetzt werden, wobei außer dem „Pause aufheben“ Knopf alle anderen Knöpfe und Tabs inaktiv werden.

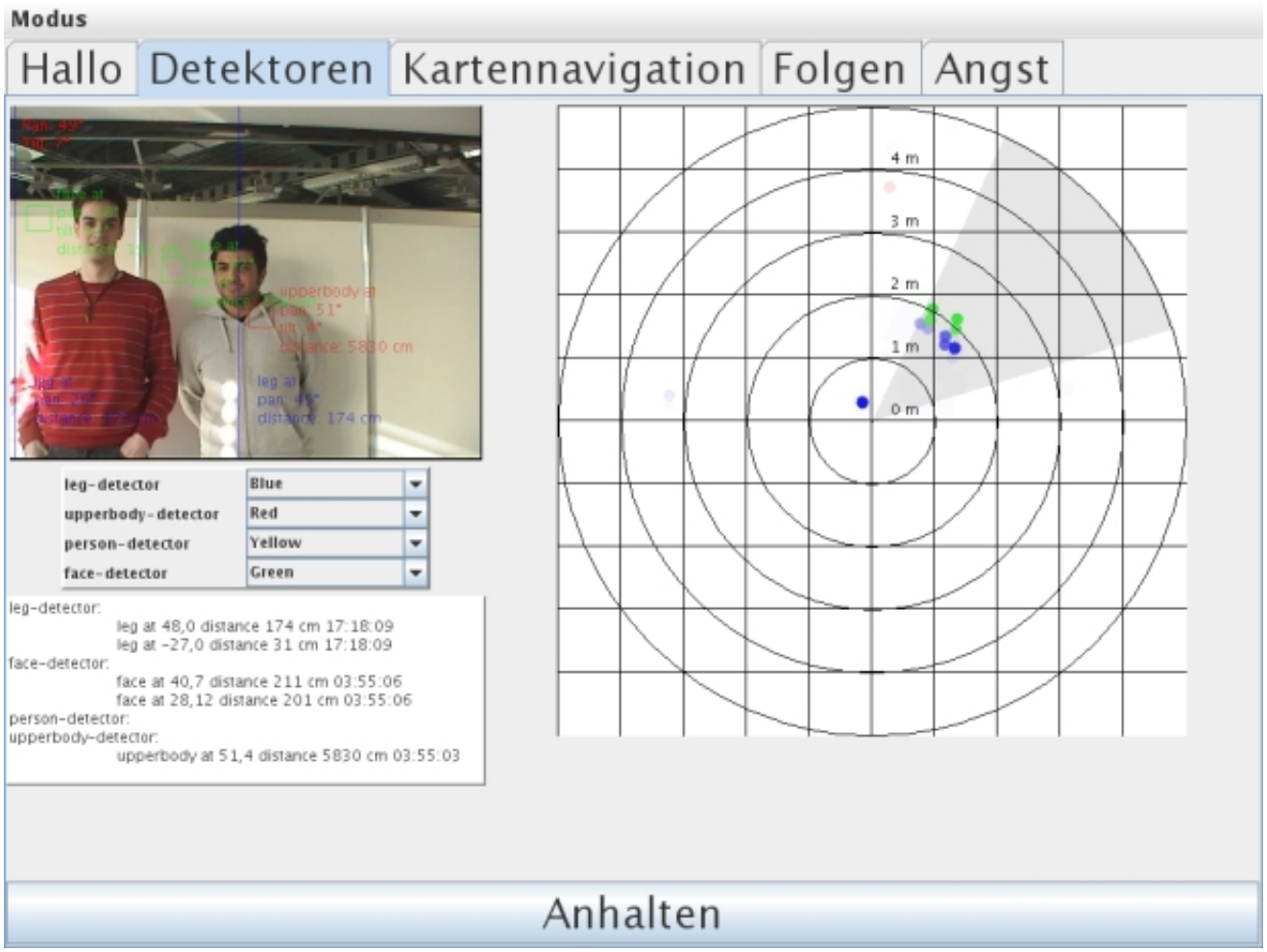


Abbildung 71: GUI Detektoren-Tab wird angezeigt, wenn sich der Roboter im Zustand „Idle“ (Personensuche) befindet

Abbildung 72: GUI Begrüßungsseite wird angezeigt, wenn der Roboter Kontakt mit Person aufnimmt

C Parameter und Skripte

C.1 Dynmap

Im Nachfolgenden sind die Parameter kurz beschrieben, die an den Dynamicmapping-Treiber mittels der Config-Datei übergeben werden können.

dislocation Entfernung, in der auf Verschiebungen getestet wird (in Pixeln, Standard 5).

functionality Gibt die Funktionalität des Treibers an (Standard 1).

- **0** Die Funktionen des Treibers sind deaktiviert. Das System verhält sich so, als würde der Treiber Wavefront seine Karte direkt vom Treiber Mapfile erhalten.
- **1** Sämtliche Funktionen des Treibers sind aktiviert.
- **2** Die unter 3.2.1 beschriebene stabile Variante, die unter der Voraussetzung funktioniert, dass nur Lasermessungen verwendet werden, wenn sich der Roboter bewegt.
- **3** Die automatische Winkelverbesserung ist aktiv, die Verschiebung der X-Y-Position wird jedoch hier nicht berücksichtigt.
- **4** Die Verschiebung der X-Y-Position wird korrigiert, der Winkelfehler wird hier nicht behandelt.

info Informationen des Treibers werden in der Konsole ausgegeben (Standard 1).

max_angle Maximale Winkeländerung, unterhalb welcher die Lasermessung noch verwendet wird (in Grad, Standard $0,15°$).

max_dist Maximale Entfernung, innerhalb der Lasermessung verwendet wird (in Metern, Standard 3,5m).

min_dist Minimale Entfernung, ab der die Lasermessung verwendet wird (in Metern, Standard 0,5m).

scan_depth Tiefe, bis zu der umliegende Pixel untersucht werden, ob sie Wände sind (in Pixeln, Standard 3).

sleeping_time Sleeping-Time des Threads (in Mikrosekunden, Standard 100000μs = 0,1s)

threshold Grenze, ab der Pixel in die dynamische Karte übernommen werden, die auch an den Treiber Wavefront übergeben werden (Standard 30).

upper_bound Grenze, bis zu der die Pixelwerte hochgezählt werden (Standard 120).

Befehl	Beschreibung
`1_N`	Ein Vollbild senden
`1_T`	Ein Vollbild mit Zeit senden
`1_V`	Ein Vollbild mit Kameraeinstellungen und Zeit senden
`1_R`	Ein Viertelbild senden
`1_U`	Ein Viertelbild mit Zeit senden
`1_W`	Ein Viertelbild mit Kameraeinstellungen und Zeit senden

Tabelle 20: Wichtige Bildabrufbefehle des Kameraservers

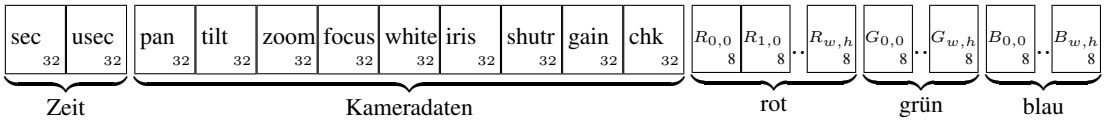


Abbildung 73: Daten vom Bildserver

D Werkzeuge und Hilfsprogramme

D.1 Kamera

D.1.1 Kameraserver

Auf dem SCITOS läuft eine von uns angepasste Variante des im IRF auch für die Finca eingesetzten „ViscaServers“. Es wurde darauf geachtet, die Kompatibilität zu existierenden Clients zu erhalten. Der Kameraserver ermöglicht den Zugriff auf die Kamera des Roboters via TCP/IP über zwei verschiedene Ports. Auf dem Steuer-Port (4000) können Visca-Kommandos zur Steuerung der Kamera übermittelt werden, auf dem Bild-Port (4500) können Bilder mit Statusinformationen abgerufen werden. Grundsätzlich sind mehrere simultane Verbindungen zum Bild-Server möglich, während der Steuer-Server nur eine simultane Verbindung erlaubt.

Der Bild-Server greift über die MetraLabs-Bibliotheken auf den DFG1394 Framegrabber zu. Es werden grundsätzlich nur zwei Auflösungen für Bilder unterstützt, PAL-Vollbilder in 768x576 und Viertelbilder in 384x288 im RGB-Modus. Viertelbilder verwenden nur jeden 2. Pixel jeder 2. Scanline und vermeiden daher Interlacing-Artefakte. Zu jedem Bild können der Zeitpunkt (Timestamp) des Abrufes sowie die grundlegenden Kameraeinstellungen Pan, Tilt, Zoom, Fokus, Weißabgleich-Modus, Iris, Verschlußzeit (Shutter) und Verstärkung (Gain) mitübertragen werden. Tabelle 20 gibt die wichtigsten Befehle wieder.

Der Server sendet beim Verbindugsaufbau zunächst den String `768_576_3_0_-1` (Breite, Höhe, Planes, Kompression aus, RGB). Als Antwort auf eine der Anfragen aus Tabelle 20 wird zunächst die Zeit, dann die Kameraeinstellungen und anschließend drei Teilbilder (zeilenweise) in rot, grün und blau (siehe Abb. 73) gesendet.

Der Steuer-Server reicht die gesamte Visca-API durch. Tabelle 21 gibt die wichtigsten Befehle wieder. Der Steuer-Server antwortet ausschließlich mit textuellen Meldungen der Form „$code\_status$ - $comment$ “. Im Erfolgsfall `10_OK` werden Werte zurückgeliefert. So hat die Antwort die From `11_#`, `12_#_#` usw. Fehlermeldungen beginnen mit `4`n`_ERROR`.

Befehl	Parameter	Beschreibung
`set_pantilt_absolute_position`	$vp\ vt\ p\ t$	Kamera zu $p\ t$ bewegen*
`get_pantilt_position`		Kameraposition abfragen
`set_pantilt_relative_position`	$vp\ vt\ \Delta p\ \Delta t$	Kamera relativ bewegen*
`set_zoom_value`	0 - 16384	Zoom setzen
`get_zoom_value`		Zoom abfragen
`set_shutter_value`	0 - 27	Verschluss (1/60 - 1/10.000)
`get_shutter_value`		Verschluss abfragen
`set_iris_value`	0 - 17	Iris setzen (Zu - F1.8)
`get_iris_value`		Iris abfragen
`set_gain_value`	1 - 7	Verstärkung (0-18dB)
`get_gain_value`		Verstärkung abfragen
`set_whitebal_mode`	0 - 3	Weißabgleich setzen
`get_whitebal_mode`		Weißabgleich erfragen
`set_focus_auto`	0 / 1	Autofokus aus/ein
`set_focus_value`	1000 - 40959	Fokus setzen
`get_focus_value`		Fokus abfragen
`memory_set`	0 - 5	Einstellungen speichern
`memory_recall`	0 - 5	Einstellungen laden
`quit`		Verbindung beenden

vt = Tilt-Geschwindigkeit (1 - 24), vp = Pan-Geschwindigkeit (1 - 24)
p = Pan-Position (-2278 - 2278 $\approx$ -170 - 170°), t = Tilt-Position (-412 - 1200 $\approx$ -30 - 90°)

Tabelle 21: Wichtige Steuerbefehle des Kameraservers

D.1.2 ViscaThere

Dieses winzige Komandozeilenprogramm überprüft ob der Kameraserver läuft.

D.1.3 Recorder

Das Recorder Programm erlaubt die Aufzeichnung längerer Bilderserien inklusive aller Statusinformationen von der Kamera.

D.2 Personendetektion

D.2.1 genhogtraindata

Das Programm dient zur Erstellung von Trainingsdateien für die FANN- oder SVM-Bibliothek. Es werden HOGs berechnet wie später im Erkenner auch. Das Programm basiert auf einer Lösung von Jan Richarz, die wir für unsere Zwecke angepasst haben.

```
genhogtraindata [options] <listfile> <outfile>
```

`--help`	Display help text.
`--verbosity #`	Set verbosity level (0 .. 10)
`--quiet`	Set verbosity level to -1, i.e. almost no output.
`--config file`	Read HOG descriptor configuration from config file.
`--split r1_r2`	Randomize data vectors into validation, train and test set. This takes 2 parameters $(0 < p < 1)$ in the form r1_r2 where the first specifies the probability for the train set and the second for the validation set. Three output files will be generated with the value of `outfile` being the prefix.
`--seed #`	Initialize Random seed to #, -1 for time.
`--output n1_n2`	Desired output values for samples. Takes 2 numbers in the form n1_n2, the first being the number of outputs and the second the one that shall be set to 1.0.
`--annotationfile file`	Annotation file with rectangular selections

D.2.2 FANNTrainer

Dieses Tool von Jan Richarz trainiert ein Multi-Layer-Perceptron mit der FANN Bibliothek.

D.2.3 svm-train

Dieses Tool ist Teil der `libsvm` und dient zum Erstellen eines SVM Modells aus Trainingsdaten.

D.2.4 PDValidate

Zur Kreuzvalidierung wurde dieses Programm entwickelt. Es wendet eine SVM oder ein KNN auf eine Menge von Eingabedaten an und gibt die Erkennungsrate und Zahlen in leicht weiterverwertbarer Form aus.

```
PDValidate [--fann|--svm] [--netfile <file>]
           [--threshold #] [--win-ratio #] <files>
```

D.2.5 HSDetect

Das Programm HSDetect führt die Erkennung mittels HOGs und einem SVM-Modell oder einem KNN sowie Mean-Shift-Clustering auf einer Liste von Bildern aus und zeichnet die Ergebnisse ein.

D.2.6 PDDemo

Diese Demoapplikation führt die Erkennung mittels HOGs und einem SVM-Modell oder einem KNN sowie Mean-Shift-Clustering auf den Livebildern vom Kameraserver durch und zeichnet die Ergebnisse ein.

```
PDDemo [--fann|--svm] [--netfile <file>]
           [--threshold #] [--win-ratio #] <files>
```

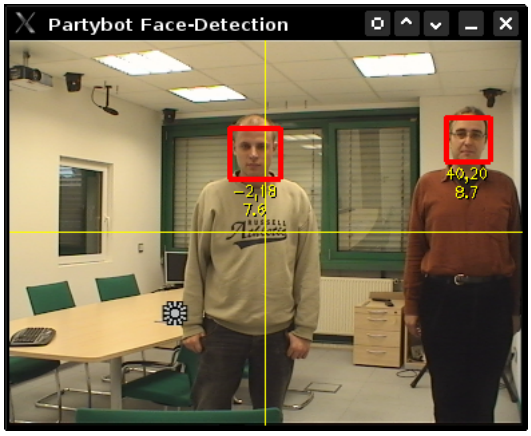


Abbildung 74: FaceDetection

D.2.7 FaceDetection

Dieses Programm wendet den VJ Gesichtserkenner auf Bilder vom Kameraserver oder von der Festplatte an (Abbildung 74).

```
FaceDetection [--cascade <haarcascade>] [--file <imagefile>]
                          [--server <server>][--port <port>]
```

D.3 Player

Das Player 2.1 Roboterframework [48] bietet, ähnlich einem Betriebssystem für Roboter, eine Hardwareabstraktion an. Damit lässt sich über eine einfache und standardisierte Schnittstelle die Hardware eines Roboters ansprechen. Player dient somit als Vermittler zwischen dem Programm und der Hardware. Das Framework selbst steht unter der GNU/GPL Lizenz und unterstützt somit durch seine quelloffenen Struktur die Implementation eigener Treiber. Des Weiteren sind bereits sehr viele Treiber und Programme für das Framework verfügbar. Das Framework ist in zwei Teile aufgeteilt, den Server, der auf dem Roboter läuft und einem oder mehreren Clients, die über eine TCP-Verbindung mit dem Server verbunden sind. Der Server kommuniziert mit der Hardware des Roboters über Treiber, bietet dem Client aber nur eine standardisierte Schnittstelle an. Einen weiteren wichtigen Aspekt für die Benutzung des Playerframeworks ist, dass es einen Simulator (Stage) gibt, mit dem die Funktionen und Algorithmen auch ohne reale Hardware getestet werden können. In Abbildung 75 ist das Zusammenspiel zwischen Programm, Player-Server und der realen Hardware illustriert.

Verwendete Treiber

Das Playerframework bietet eine Vielzahl unterschiedlicher Treiber. In Abbildung 76 sind die für dieses Projekt relevanten Treiber illustriert. Des Weiteren ist die Interaktion zwischen den einzelnen Treiber des Playerframeworks veranschaulicht. Nähere Informationen zu den in diesem Projekt verwendeten Treiber werden im Folgendem näher erläutert.

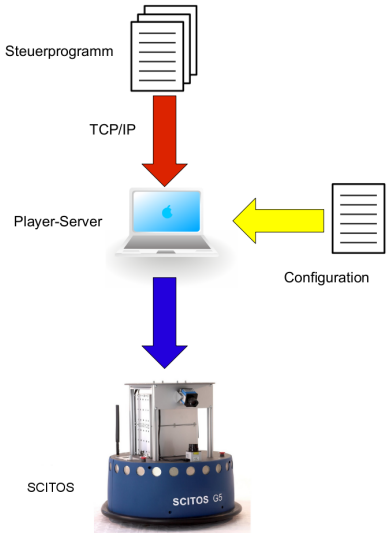


Abbildung 75: Player-Server mit realem Roboter

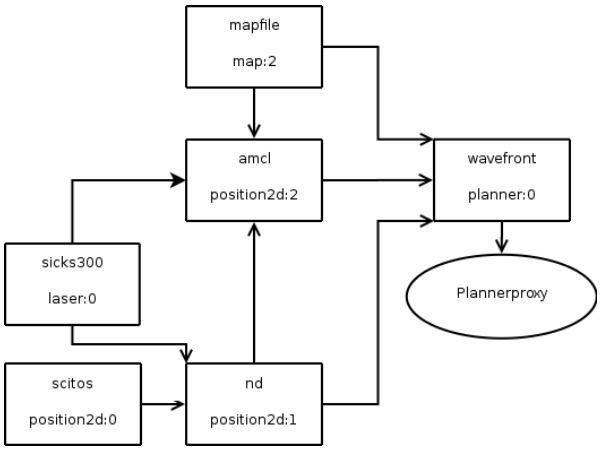


Abbildung 76: Treiber im Playerframework

AMCL - Die Adaptive Monte-Carlo Lokalisation dient zur Lokalisierung des Roboters anhand der übergebenen Karte. Als Ergebnis werden Hypothesen für die wahrscheinlichste Position ausgegeben.

Dynamicmapping - Der im Rahmen der Projektgruppe entwickelte Treiber zur dynamischen Hinderniserkennung, siehe Seite 37.

Mapfile - Erstellt aus einer PNG-Datei eine Karte, die an andere Treiber zur Lokalisation oder Navigation weitergereicht wird.

ND - Die Nearness Diagramm Navigation verhindert Kollisionen mit Objekten. Sie lässt den Roboter Hindernisse umkreisen oder löst einen Notstopp aus, wenn der Roboter ein Hindernis zu rammen droht. Dieser Treiber löst jedoch keine Neuplanung aus, weswegen wir einen neuen Treiber entwickelten.

SCITOS - Die reale Hardware des SCITOS. Sie stellt Odometriewerte zur Verfügung und nimmt Fahrkommandos entgegen.

Sicks300 - Treiber zur Ansteuerung des Sicks 300 Laser. Es handelt sich um eine innerhalb des Institutes für Roboterforschung angepasste Version des Sicks3000 Treibers.

Wavefront - Pfadplanungsalgorithmus basierend auf dem Wavefront-Algorithmus [30, S.377 ff].

D.4 Diverse

D.4.1 EBCTool

Dieses kleine Programm schaltet die EBC-Ports des SCITOS ein oder aus.

```
ecbtool [--port <port>] [--voltage <voltage>]  [--off|--on]
```