Institut für Informatik VI
Abteilung Autonome Intelligente Systeme
Universität Bonn
Friedrich-Ebert-Allee 144
53113 Bonn

Teleoperation eines Service-Roboters mit Android-Handhelds

Sebastian Muszynski

Studiengang:	Informatik
Erstgutachter:	Prof. Dr. Sven Behnke
Zweitgutachter:	Prof. Dr. Armin B. Cremers
Betreuer:	Jörg Stückler
begonnen am:	06. September 2011
beendet am:	16. April 2012

Inhaltsverzeichnis

1 Einleitung

Schon heute ist ein Trend erkennbar, welcher weg von den Industrie-Robotern zu persönlichen Service-Robotern führt. Mit zunehmender Entwicklung ihrer Fähigkeiten wird sich die Zweckgebundenheit von Robotern auflösen. Sie werden eine wachsende Zahl an Aufgaben beherrschen und einen vielseitigen Einsatz finden. Die Mensch-Roboter-Interaktion, eine spezielle Form der Mensch-Maschine-Interaktion, rückt deshalb zusehends in das Interesse und den Vordergrund der Robotik. Diese Arbeit befasst sich im weiteren mit der Mensch-Maschine-Interaktion im Bereich der persönlichen Service-Robotik.

Roboter lassen sich in drei Klassen einteilen: Industrie-Roboter, Roboter zur Erfüllung ganz spezieller, meist einmaliger Aufgaben und Service-Roboter.
Industrie-Roboter kommen in modernen Fertigungsanlagen zum Einsatz, sind weniger intelligent und erfüllen feste Anforderungen. Die Tätigkeit, die sie ausüben wird fortwährend wiederholt. Da Industrie-Roboter für einzelne, sehr spezielle Aufgaben eingesetzt werden, ist eine starke Optimierung möglich. Die Präzision steht im Vordergrund, wobei die Mobilität und damit auch ihr Gewicht eine untergeordnete Rolle spielen.
Service-Roboter stellen das direkte Gegenstück zu Industrie-Robotern dar. Sie müssen in der Lage sein, sich einer dynamischen Umgebung anzupassen und unterzuordnen. Mobilität erlaubt dem Roboter sich in dieser Umgebung zu bewegen. Diese Anforderungen führen zu einem intelligenten Verhalten. Die Präzision, mit welcher Aufgaben gelöst werden, kann vernachlässigt werden, sofern das gewünschte Ziel erreicht wird. Mit zunehmender Entwicklung werden Service-Roboter eine Vielzahl von Aufgaben beherrschen und dem Menschen immer ähnlicher werden. Haushaltsroboter sind im Fachhandel heute schon erhältlich und stehen dem Menschen in speziellen Aufgaben zur Seite. Die Befehle für diese speziellen Aufgaben sind bisher einfach und direkt: „Putze jeden Dienstag um 10 Uhr“. So lautet die Arbeitsanweisung für einen Reinigungsroboter der Firma iRobot. Das Bedienkonzept angelehnt an eine Zeitschaltuhr erlernt der Anwender schnell. Die Aufgabe und das Ziel ist klar definiert. Der Putzvorgang erfolgt daraufhin in vollständiger Autonomie. Das der Roboter dabei seine Umgebung nicht verwüstet, sich selbst keinen Schaden zufügt, indem Treppenstufen gemieden werden und zum Ende seiner Arbeit die Ladestation aufsucht, wird schnell zur Selbstverständlichkeit.
Der Spionageroboter WowWee der Firma Rovio [Wow] ist über eine Web-Oberfläche aus einem Browser heraus bedienbar. Wird er nicht durch einen virtuellen Joystick teleoperiert, so patroulliert er auf Wunsch auf einer zuvor definierten Folge von Wegpunkten und überträgt dabei Kamerabild und die Aufnahme des integrierten Mikrofons.

Die Konsequenz ist, dass sobald Roboter bei zunehmender Leistungsfähigkeit vollständig in den menschlichen Alltag integriert werden, eine einfache Benutzerschnittstelle zur effizienten Interaktion unumgänglich ist. Ausschlaggebend

ist ein abstrakter Informationsaustausch, um Geschwindigkeit und damit einen echten Nutzen zu erhalten.
In der RoboCup@Home-Liga, in welcher der aktuelle Stand der Forschung der Service-Robotik jährlich präsentiert wird, schreibt das Regelwerk die natürliche Sprache zur Interaktion mit dem Roboter vor. Da es sich um eine umfassend definierte Testaufgabe handelt, die vollständig autonom gelöst werden soll, ist dieser Ansatz für einen Wettbewerb und das Messen von Performanz und Fähigkeiten des Roboters vorteilhaft. Menschliche Einflüsse während des Betriebs würden das Ergebnis verfälschen und die Bewertung erschweren.
Geht man nun aber davon aus, dass nicht jedes Detail einer Aufgabe in gesamter Vollständigkeit vorab definiert werden kann, so erweist sich die natürliche Sprache schnell als unvorteilhaft. Wird ein Kind aufgefordert einen Gegenstand zu bringen, von welchem es keine Vorstellung besitzt, so wird versucht sich mit Hilfe einer Umschreibung des Gegenstandes zu behelfen oder den Aufenthaltsort genau einzugrenzen. Das Kind ist damit in der Lage Schlussfolgerungen zu ziehen und mit Hilfe ausgeprägter adaptiver und kognitiver Fähigkeiten eine Entscheidung zu treffen. Dennoch ist die erfolgreiche Ausführung der Aufgabe nicht sichergestellt. Das Kind wird häufig unverrichteter Dinge zurückkehren. Ein neuer Versuch wird unternommen. Dieses Mal wird dem Kind der Gegenstand gezeigt, um einen Lerneffekt für die Zukunft zu erreichen. Diese komplexen Zusammenhänge von Assoziationen, Intuition und erlerntem Wissen sind bis heute nicht ausreichend erforscht, um sie künstlich nachzubilden. Auch der beherrschte Wortschatz und die einzelnen Facetten einer Aktivität des Roboters ist für den Anwender schwer zu bestimmen.

Um die Fähigkeiten eines Dienstleistungsroboters uneingeschränkt nutzen zu können, wird in dieser Arbeit ein Handheld zur Interaktion mit dem Roboter eingesetzt. Mit Hilfe drahtloser Kommunikation ist es möglich mit dem Roboter aus der Ferne zu interagieren. Das primäre Ziel der Teleoperation eines persönlichen Service-Roboters ist nicht die kontinuierliche Steuerung, mit welcher der Roboter zu einem Ort geführt wird oder eine Greifbewegung Gelenk für Gelenk ausführt. Vielmehr soll der Dienstleistungsroboter weitgehend autonom agieren und definierte Aufträge entgegen nehmen. Bei schwierigen nicht autonom lösbaren Aufgaben soll der Roboter dem Anwender bei der Steuerung assistieren und vor Gefahren schützen. Im Falle eines Fehlers wird der Anwender informiert. Auch bei komplexen Entscheidungsprozessen kann die Aufmerksamkeit des Anwenders nützlich sein, um dem Roboter unterstützend zur Seite zu stehen. Denn der Mensch ist bis heute der bessere Entscheider auf Grund von Intuition und Lebenserfahrung.

Die Unterstützung und Entlastung des Anwenders steht dabei im Vordergrund. Erst wenn ein Roboter eine Tätigkeit effizienter ausübt als ein Mensch oder der Anwender parallel anderen Aktivitäten nachgehen kann, ist ein signifikanter Nutzen feststellbar.
Aus diesen Gründen entwickelt sich die Mensch-Roboter-Schnittstelle zunehmend zu einer kritischen Komponente [SCG09]. Um eine hohe Gesamtperformanz bei

Systemen zur Navigation und Manipulation zu erreichen, ist ein intuitives und für den Alltagsgebrauch kompaktes Steuerinstrument notwendig.
Die Integration von Robotik-Anwendungen in Haushalten wird durch bereits etablierte Technologien zunehmend einfacher [PLKB08].
Ein Beispiel dafür ist der Standard für drahtlose Netzwerke (WLAN). Kaum eine Wohnung ist nicht mit einem drahtlosen Netzwerk ausgestattet. Gerätehersteller rechnen damit, dass diese Technologie für den Einsatz ihres Produktes zur Verfügung steht, ebenso wie in der eigenen Wohnung eine Verbindung zum Internet als selbstverständlich angesehen wird. Auch der zunehmende Einsatz von intelligenten Anwendungen wird die Integration vereinfachen. Ein Roboter wird in der Lage sein Haustechnik zu steuern, den Stromzähler abzulesen oder Multimedia-Systeme zu kontrollieren.

1.1 Steuerinstrument

Obwohl schon in den neunziger Jahren erste Anwendungen für mobile Geräte, den sogenannten Handhelds, angeboten wurden, war deren Akzeptanz äußerst gering. Die Ursachen sind vielfältig. Die Bildschirme der Geräte waren überdurchschnittlich klein, der verfügbare Arbeitsspeicher stark limitiert und die Rechenleistung niedrig. Das Erreichen einer hohen Akkulaufzeit bei kompakten Ausmaßen stand im Vordergrund. Auch Internetdienste waren verfügbar, jedoch lieferte das Mobilfunknetz nur eine sehr geringe Bandbreite. Die angebotenen Dienste mussten optimiert werden. Die Nutzung einer mobilen Datenverbindung war kostspielig, sie wurden ebenso wie Gesprächsminuten nach einer Verbindungsdauer abgerechnet. Volumentarife waren kaum verbreitet und wurden nur in Kombination mit einem Mobilfunkvertrag über die gesamte Vertragslaufzeit angeboten.

Ein weiteres Problem bestand in der starken Fragmentierung der Laufzeitumgebungen. Nahezu jeder Hersteller pflegte ein eigenes Betriebssystem und damit eine eigene Laufzeitumgebung. Zur Lösung dieses Problems wurden Geräte mit einer besonderen Java-Plattform ausgerüstet. Die Plattform trägt den Namen „Java Micro Edition“ (J2ME) und sollte die Anwendungsentwicklung für bestimmte Geräte-Klassen erleichtern.

Bis heute hat ein grundlegender Wandel stattgefunden. Der Markt der mobilen Geräte hat in den vergangenen Jahren ein enormes Wachstum erfahren. Die durchschnittliche Bildschirmgröße hat zugenommen und ein Farbbildschirm ist selbstverständlich. Berührungsempfindliche Bildschirme werden eingesetzt, um die Bedienung so einfach wie möglich zu gestalten. Besondere Stifte zur präzisen Interaktion mit den Geräten sind nicht mehr notwendig.
Auch die Mobilfunknetze haben eine Entwicklung erfahren. Neue Technologien liefern ausreichend Bandbreite für den mobilen Zugriff auf das Internet. Volumenpakete der Mobilfunkanbieter genießen eine breite Akzeptanz. Trends und

Hersteller haben die Entwicklung voran getrieben. Die Java Micro Edition gehört ebenfalls der Vergangenheit an. Sie wurde gegen die Standardversion, wie sie auch auf einem normalen Computer zum Einsatz kommt, ersetzt. Daran ablesbar ist der enorme Ressourcenzuwachs den die Handhelds erfahren haben.

Die Anwendungsentwicklung für mobile Geräte unterscheidet sich dennoch weiterhin von der Entwicklung einer Desktopanwendung. Es darf nicht vernachlässigt werden, dass die Geräte eine eigene unabhängige Stromversorgung besitzen und handlich sein sollen. Die geringen Ausmaße werden die Interaktion und die Darstellung immer beeinflussen. Eingesetzte Hardware-Tastaturen sind ebenfalls klein, wenn überhaupt vorhanden. Eingaben über den Bildschirm können effektiv, aber auch kompliziert und ungenau sein. Häufig handelt es sich um leistungsstarke Mobiltelefone, so genannte Smartphones. Es muss zusätzlich Sorge getragen werden, dass ein Endgerät nicht so stark belastet wird, dass kein Anruf mehr entgegen genommen werden kann. Kommt ein Anruf herein, so muss eine Anwendung angemessen reagieren und zum Beispiel Schlafen gelegt werden. Speicherlecks oder eine unsaubere Programmierung darf nicht das gesamte Gerät zum Absturz bringen.

Die heutige Generation der Smartphones und Tablets mit Android-Plattform sind ein gutes Instrument, um eine Teleoperationsanwendung für Roboter zu ermöglichen. Sie genießen eine starke Verbreitung und eine gut zugängliche Plattform zur Entwicklung neuer Anwendungen. Des weiteren stellen sie ausreichend Rechenleistung und Arbeitsspeicher zur Verfügung. Ein berührungsempfindlicher Bildschirm kommt zur einfachen und geräteübergreifenden Interaktion zum Einsatz.

1.2 Teleoperation

Die Anwendungsbereiche für Teleoperation bei autonomen Robotern sind vielfältig. Die aktuelle Forschung beschäftigt sich hauptsächlich mit Aktivitäten, die ein Anwender selbst nicht ausführen kann oder soll. Gründe dafür sind eingeschränkte Mobilität, die Bindung an ein Krankenbett oder im Katastrophenszenario die Erkundung von gefährlichen Umgebungen. Bei steigender Gesamtperformanz und sinkenden Kosten werden die Einsatzgebiete zunehmen.

Ein kompaktes und intuitiv bedienbares Steuerinstrument ist in jedem Fall wünschenswert. Das Erlebnis des Anwenders wird verstärkt, wenn das zur Steuerung eingesetzte Gerät bereits einen festen Platz im Alltag eingenommen hat. Der Anwender ist mit dessen Bedienung vertraut und ist mit ihm in der Lage dem Roboter Anweisungen aus der Entfernung zu erteilen. Im Katastrophenfall und unter Stress wirkt sich ein vertrautes Werkzeug ebenfalls positiv aus.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung einer Android-Anwendung, mit welcher ein Service-Roboter auf drei unterschiedlichen Autonomiestufen teleoperiert werden kann. Dafür stehen Roboter und die Anwendung dauerhaft über eine drahtlose Netzwerkverbindung in Kontakt. Es ist möglich aus der Entfernung den Roboter über ein Livebild oder die aktuelle Position in einer Karte zu lokalisieren.

Entworfen wird ein Anwendungsszenario mit verschiedenen Schwierigkeitsgraden, welches in den unterschiedlichen Autonomiestufen lösbar ist. Je höher die Autonomie, desto weniger Interaktion wird vom Anwender gefordert. Geschaffen werden auf diese Weise drei Abstraktionsschichten, bestehend aus intuitiven Bewegungskonzepten, verketteten aber nicht weiter unterteilbaren Aktionsprimitiven und vollständigen Aktivitäten mit semantischen Zielen.

1.4 Anwendungsszenario

Um die Aufgabe der Anwendung besser zu illustrieren wird im folgenden ein Beispielszenario definiert. Die Benutzerrolle nimmt eine in seiner Mobilität stark eingeschränkte Person ein. Diese ist an ihr Bett gebunden und körperlich nicht in der Lage dieses zu verlassen. Ein Szenario, welches häufig im Alter anzutreffen ist oder bei Menschen, die Opfer einer schweren Krankheit oder eines Unfalls geworden sind. Die Person wird in einer gewohnten Umgebung von ihren Angehörigen und durch Unterstützung eines Pflegedienstes versorgt. Für jede Aktivität, die nicht aus der Reichweite des Betts zu bewerkstelligen ist, benötigt die Person fremde Hilfe.
Der persönliche Service-Roboter soll als Hilfsmittel dienen, um die Abhängigkeit von Anderen zu mindern und die Lebensqualität zu verbessern. Der Anwender erlangt auf diese Weise einen Teil seiner Selbstständigkeit zurück. Der Roboter soll Aufgaben ausführen, welche dem Anwender auf Grund seiner eingeschränkten Mobilität sonst nicht möglich sind. Die Handlungsfreiheit des Anwenders soll dabei nicht durch die Autonomie des Roboters eingeschränkt werden. Die für diese Arbeit gewählte Beispielaufgabe ist das Holen eines gewünschten Objektes aus einem entfernten Raum. Der Anwender hat zum Zeitpunkt der Identifikation des Objekts und dem Greifvorgang keine direkte Sicht auf die Szene.
Ein Android-Tablet dient zur Interaktion mit dem Roboter und sorgt für einen Einblick in die Szene. Das Gerät stellt die Schnittstelle zwischen Roboter und Anwender zur Verfügung. Eine drahtlose Netzwerkverbindung zur Kommunikation ist zwingend notwendig. Natürliche Sprache kommt in diesem Anwendungsszenario nicht zum Einsatz. Auch die Tatsache, dass alte Menschen für gewöhnlich nur schlecht Sehen und ihr Koordinationsvermögen, insbesondere wenn sie bereits ans Bett gebunden sind, stark beeinträchtigt ist, soll an dieser Stelle vernachlässigt werden.

1.5 Gliederung der Arbeit

Kapitel 2 vermittelt die für diese Arbeit notwendigen Grundlagen. Eingegangen werden auf wichtige Aspekte der Mensch-Roboter-Interaktion, die bei der Entwicklung der Anwendung eine entscheidende Rolle einnehmen. Es werden die Eigenschaften der Android-Plattform und der Entwicklung von Anwendungen für dieses Betriebssystem erläutert, sowie die auf dem Roboter eingesetzte Middleware und verwendete Schnittstelle zum Nachrichtenaustausch. Anschließend erfolgt in Kapitel 3 eine Vorstellung und Diskussion von Arbeiten, die sich mit verwandten Problemstellungen beschäftigen. In Kapitel 4 wird der Entwurf der Anwendung beschrieben, die eingesetzten Komponenten erläutert und Entwurfentscheidungen begründet. Kapitel 5 zeigt interessante Gesichtspunkte der Implementierung auf. Ausgewählt wurde das Verfahren zur Erzeugung einer erweiterten Realität und die folgende Selektion virtueller Objekte. Die darauf folgenden Abschnitte widmen sich in Kapitel 6 der Bewertung der entwickelten Anwendung an Hand durchgeführter Experimente im Rahmen einer Benutzbarkeitsstudie. Kapitel 7 fasst die gewonnen Erfahrungen im Laufe der Entwicklungs- und Testphase zusammen und gibt Ausblicke, wie die Arbeit thematisch fortgeführt werden kann.

2 Grundlagen

2.1 Service-Robotik

Die Service-Robotik umfasst alle Robotik-Anwendungen, die Dienstleistungen erbringen und den Menschen bei definierten Aufgaben teil- bis vollautomatisch unterstützen. Dienstleistungensroboter zeichnen sich aus durch Mobilität, der eigenständigen Ausübung von Tätigkeiten und der Manipulation ihrer Umgebung. Das Themengebiet der Service-Robotik unterteilt sich in zwei Bereiche: Die professionelle und die persönliche Service-Robotik.
Die professionelle Service-Robotik kommt in Situationen zum Einsatz, bei denen ein Mensch seine Gesundheit gefährden würde. Dazu gehört die Beseitigung von Atommüll, das Entschärfen von Bomben oder die Suche nach Überlebenden in einem Erdbebengebiet. Aber auch in anderen Disziplinen kommt die professionelle Service-Robotik zum Einsatz. Es gibt Dienstleistungsroboter, die auf Grund ihrer Präzision in der Medizin als Operationsassistent eingesetzt werden oder im Zuge ihrer kompakten Größe zur Inspektion von Kanalrohren behilflich sind.
Nach einer Studie [oR] der Vereinten Nationen und der International Federation of Robotics über die Einsatzgebiete der Service-Robotik hat sich die Anzahl der professionellen Service-Roboter im Zeitraum von 2001 bis 2005 verdoppelt.

Die verwendeten persönlichen Service-Roboter hingegen haben in diesem Zeitraum eine beachtliche Steigerung von 1.145 Prozent erfahren.
Persönliche Service-Roboter gehen Menschen innerhalb ihrer Wohnung zur Hand. Sie unterstützen oder assistieren bei regelmäßig wiederkehrenden Aufgaben. Häufige Anwendungsgebiete sind das Staubsaugen, Rasenmähen, Fensterputzen oder die Schwimmbadreinigung. Zur Unterstützung von älteren oder körperlich behinderten Menschen kommen persönliche Service-Roboter ebenfalls zum Einsatz. Auch kann manches Spielzeug aus der Unterhaltungselektronik zu den persönlichen Service-Robotern gezählt werden.

Somit ist die Mehrheit der Dienstleistungsroboter nicht mehr professioneller Natur, sondern im persönlichen Umfeld anzutreffen. Thrun [Thr04] beschreibt, dass sich durch diese Verschiebung neue Anforderungen und Zielsetzungen für den Roboter und damit auch der Interaktion mit dem Roboter ergeben:
Roboter teilen sich nun ein gemeinsames Umfeld mit dem Menschen. Zur Bedienung des Roboters kommen keine Spezialisten mehr zum Einsatz. Die Steuerung ist stark von der Aufgabe des Roboters abhängig und erfordert eine genaue Abstimmung auf den Benutzer.
Der Roboter muss neuen Sicherheitsanforderungen entsprechen. Er soll weder dem Menschen, seiner Umwelt, noch sich selbst Schaden zufügen. Trotz sich stetig ändernder Umgebung hat er robust, möglichst genau und zuverlässig zu arbeiten. Auch der Kostenfaktor spielt eine entscheidende Rolle.

2.2 Robotersteuerung

So unterschiedlich verschiedene Roboter und ihre Einsatzgebiete sind, so unterschiedlich sind auch die Methoden zur Steuerung. Das Spektrum erstreckt sich von der direkten Steuerung bis hin zur vollständigen Autonomie. Ist ein Roboter autonom, so besitzt er die Fähigkeit Aufgaben selbstständig zu lösen. Der Anwender erteilt dazu einen Auftrag, den der Roboter ausführt. Bei Problemen kann der Anwender als Entscheider zu Rate gezogen werden. Umso größer die Arbeitserleichterung, desto höher ist der Nutzen für einen Anwender. Die direkte Steuerung hingegen beschreibt die zeitlich direkte und kontinuierliche Bedienung eines Roboters. Sie kommt oft zum Einsatz, wenn Autonomie keine robuste Lösung für einen Anwendungsfall liefert.

Diese Methoden haben unterschiedliche Anforderungen an die Robotersteuerung und die Aufgaben, die der Roboter übernimmt. In jedem Fall ist es von Vorteil, wenn der Anwender einen Einblick in die Szene erhält, in welcher der Roboter agiert.

2.2.1 Autonomie

Drury [Dru04] misst Autonomie oder den Grad einer Automation eines Roboters, indem er die Häufigkeit an notwendiger Assistenz beim Lösen einer Aufgabe messen. Allgemein lässt sich Autonomie auch als die selbstständige Ausführung einer komplexen Handlung beschreiben. Ein Beispiel für eine Aufgabe ist, den Roboter anzuweisen, eine Flasche Wasser zu holen. Der Roboter muss dazu wissen, was eine Wasserflasche ist, wo sie sich befindet und anschließend planen wie er von seiner Position zu der Flasche gelangt, diese greift und zum Anwender bringt.
Im Bezug auf eine Anwendung lässt sich Autonomie auch durch die Zeit beschreiben, in welcher ein Roboter vernachlässigt werden kann ohne das die Performanz einbricht. Daher wird Autonomie von Crandall und Goodrich [CG01] als Vernachlässigungstoleranz bezeichnet. Eine formale Definition von Autonomie findet sich bei [Alb91], [Bil00], [GG08].

Autonomie wird erreicht durch die Kombination von Technologien aus der Signalverarbeitung, Kontrolltheorie und der künstlichen Intelligenz. Sie ermöglicht einen abstrakten Informationsaustausch. Dabei kann es sich um natürliche Sprache handeln, aber auch um einen schmalbandigen digitalen Kommunikationsweg. Die Autonomie eines Roboters kann darüber hinaus unterschiedlich stark ausgeprägt sein. Um die verschiedenen Stufen oder Grade zu benennen hat Sheridan [She92b] eine Skala eingeführt.

Vollständige Steuerung Der Roboter wird vollständig über Aktionen des Anwenders gesteuert. Es werden keine Aktionen ausgeführt, die nicht unmittelbar vom Anwender ausgelöst wurden. Der Roboter besitzt somit keinerlei Autonomie. Häufig wird dieser Autonomiegrad als Teleoperation bezeichnet.

Aufgeteilte Steuerung In dieser Stufe nehmen sowohl der Anwender als auch der Roboter auf das Verhalten des Roboters Einfluss. Unterschieden wird in drei Arten der geteilten Steuerung:

Sichere Teleoperation Bei der sicheren Teleoperation besitzt der Roboter Schutzmechanismen, welche die Gefahren durch eine fehlerhafte Bedienung verringern. Eine Beschädigung des Roboters wird vermieden oder wenigstens gemindert.
Entworfen wurde die sichere Teleoperation zur Steuerung für Erkundungsroboter bei Mond-Missionen. Durch die hohe Verzögerungen bei der Kommunikation zwischen Mond und Erde steigt die Gefahr einer fehlerhaften Bedienung. Der Anwender neigt dazu falsche Entscheidungen zu treffen, da Informationen längst veraltet sind, wenn sie auf der Erde eintreffen.
Auch das Umfahren eines vom Anwender ungesehenen oder ignorierten Hindernisses wird in diesem Modus oft angeboten.

Teilweise Autonomie Der Roboter ist in der Lage Aktionen selbstständig auszuführen und Entscheidungen zu treffen. In bestimmten Situationen kann der Roboter Assistenz anfordern, sei es weil er eine Gefahr vermutet, einen Gegenstand nicht identifizieren kann oder auf Grund eines Engpasses keinen Weg zum Ziel findet. Der Anwender kann den Roboter bei Entscheidungen unterstützen.

Hochgradige Autonomie Der Roboter agiert weitestgehend autonom. Wenn überhaupt, dann erfolgt das Eingreifen des Anwenders auf einer sehr abstrakten Ebene.
Zu dieser Stufe zählt auch die kollaborative Steuerung. Menschen agieren dabei als Ressource des Roboters. Der Roboter erfordert keine dauerhafte Aufmerksamkeit und fordert lediglich bei Bedarf Unterstützung an. Dieses Konzept ist ideal für die Steuerung von mehreren Robotern gleichzeitig oder zur Entlastung eines einzelnen Anwenders.

Vollwertige Autonomie bedeutet, dass ein Roboter eine Aufgabe lösen kann, ohne dass ein Mensch involviert wird.

Anpassbare Autonomie oder auch gleitende Autonomie sind Konzepte, bei denen der Anwender gezielt einen Autonomiegrad für einen bestimmten Zeitpunkt wählt. Gleitende Autonomie beschreibt aber auch die Möglichkeit des Roboters, einen Menschen um Rat zu fragen oder auf andere Weise auf menschliche Interventionen zu reagieren. Den Rest der Zeit arbeitet der Roboter autonom [DKB+08].

2.2.2 Teleoperation

Der Ansatz der vollständigen Autonomie eines Roboters besitzt bis heute verschiedene Nachteile. Das Wahrnehmungsvermögen eines Roboters und die Fähigkeit korrekte Entscheidungen zu treffen kann sich bis heute nicht mit den

Fähigkeiten eines Menschen messen. Menschen treffen Entscheidungen mit Hilfe von Erfahrung oder Intuition. Deshalb ist es in vielen Fällen vorteilhaft, wenn Mensch und Maschine zusammen arbeiten und sich gegenseitig ergänzen. Erfordert ein Roboter keine fremde Unterstützung mehr, so wird die Teleoperation zusehens in den Hintergrund rücken.

Teleoperation kann verschiedenartig definiert werden. [Mur00] beschreibt die Teleoperation als die Steuerung einer Maschine über eine Distanz. Teleoperation beinhaltet außerdem, dass der Roboter sich allgemein nicht in Sichtweite des Anwenders befindet. Die Maschine stellt somit eine Erweiterung der sensorischen und manipulatorischen Fähigkeiten einer Person dar.
Die teleoperierte Maschine muss dazu Sensoren, Aktuatoren, sowie einen multimodalen Kommunikationskanal zum Anwender besitzen. Über die Aktuatoren ist es möglich eine physikalische Kraft auszuüben oder mechanische Arbeit zu vollrichten.

Die zu überbrückende Distanz spielt eine entscheidende Rolle, da die Lichtgeschwindigkeit in der Telekommunikation einen limitierenden Faktor darstellt. Umso größer die Entfernung, desto höher die Verzögerung in der Übertragung. Die Steuerung einer Marssonde, bei der die Übertragung eines Befehls (ohne Warten auf eine Reaktion) fünf Minuten benötigt, unterscheidet sich deshalb wesentlich von der Steuerung eines Retterungsroboters in wenigen Kilometern Entfernung.

Eine Sensorik ist notwendig, um dem Anwender einen Eindruck der Umgebung des Roboters übermitteln zu können. Zum Beispiel können die Sensordaten dem Anwender auf einem Monitor dargestellt werden. Als Steuerungsmechnismus werden Joysticks, Computer-Mäuse, Tastaturen oder andere innovative Eingabegeräte eingesetzt. Der Anwender wird somit in den Entscheidungsprozess eingebunden.

Teleoperation ist vorteilhaft für komplizierte Aktivitäten, die selten wiederholt werden. Für sich oft wiederholende und langweilige Aufgaben ist die kontinuierliche Steuerung aus der Ferne ungeeignet, da die Aufmerksamkeitsspanne eines Anwender begrenzt ist; die Fehleranfälligkeit wächst mit der Nutzungsdauer.

Grenzt man den Begriff der Teleoperation auf die Robotik ein, so spricht man von Telerobotik. In der Telerobotik besitzt der Anwender in vielen Fällen ausschließlich die Aufgabe eine Aktivität des Roboters zu überwachen. Es können Ziele formuliert werden oder Entscheidungen an einen zwischengeschalteten Computer übertragen werden. Die direkte Steuerung des Operators übernimmt der Computer.

In einem Standardwerk der Robotik [DN90], finden sich folgende Punkte für Charakteristiken von Anwendungen bei denen der Ansatz von Teleoperation sinnvoll erscheint:

- Die Aufgabe muss unstrukturiert und nicht-repetitiv sein.
- Der Einsatz von Industrie-Robotern ist nicht möglich, da sich die Arbeitsbedingungen laufend verändern.
- Hauptbestandteile der Aktivität erfordern geschickte Manipulation (Hand-Augen-Koordination).
- Hauptbestandteile der Aktivität erfordern Objekt-Erkennung, Situationsbewusstsein oder andere erweiterte Wahrnehmung.
- Es steht genügend Bandbreite zur Verfügung und die Latenz stellt kein Problem dar.
- Die Verfügbarkeit von trainiertem Personal ist kein Problem.

2.2.3 Telepräsenz

Neben der Teleoperation gibt es noch die Telepräsenz, hierbei steht nicht das Agieren in der Ferne im Vordergrund, sondern die Wahrnehmung einer entfernten Umgebung. Bei Telepräsenz soll dem Bediener das Gefühl vermittelt werden in der entfernten Umgebung präsent zu sein.
Steuer [Ste92] und Nielsen et al. [NGR07] definieren Telepräsenz als das Verstehen einer Umgebung, an der man selbst nicht körperlich anwesend ist. Diese Definition ist weniger restriktiv als die von Sheridan [She92a], welcher fordert, dass das Gefühl entstehen soll, dass man in der Ferne präsent ist.

Eine entscheidende Rolle spielt dabei die Immersion, ein Maß für die realistische Repräsentation der Realität. Es handelt sich nicht nur um ein Konzept aus der virtuellen Realität, sondern lässt sich auch in der Mensch-Roboter-Interaktion nutzen. Das Empfinden der Präsenz hat einen positiven Einfluss auf die Performanz. Riva et al. [RDIZ03] empfiehlt die Messung von Präsenz mit folgenden Methoden:

Subjektiv Der Benutzer wird über sein Empfinden befragt.

Verhaltensbezogen Die Präsenz wird am Verhalten des Anwenders gemessen, während er das System nutzt. Häufig werden virtuelle Objekte in Richtung des Benutzers geworfen, um dessen Reaktion zu testen.

Physiologisch Der Herzschlag des Anwenders wird gemessen, die Leitfähigkeit der Haut oder die Hauttemperatur. Diese Faktoren stehen im Zusammenhang mit der Präsenz des Benutzers.

2.3 Mensch-Roboter-Interaktion

Die Mensch-Roboter-Interaktion im englischen Human-Robot Interaction (HRI) ist das Forschungsgebiet, welches sich mit der Interaktion zwischen einem Roboter und dessen Anwender beschäftigt. HRI beschreibt den Prozess der Zusammenarbeit, um

ein Ziel zu erreichen. Motiviert durch das Näherrücken des universell einsetzbaren Roboters werden die Anwendungsszenarien zunehmend komplexer.
Solange Roboter als besseres Werkzeug für die Ausführung einer einzelnen Aufgabe entwickelt und eingesetzt wurden, war eine ausgiebige Interaktion nicht notwendig, geschweige denn erwünscht. Zum Beispiel soll ein Reinigungsvorgang vollständig autonom und möglichst in Abwesenheit des Besitzers stattfinden. Mit zunehmender Funktionalität wird eine Interaktion mit dem Roboter unumgänglich.

Es genügt nicht, wenn Service-Roboter grundsätzlich in der Lage sind eine Tätigkeit zufriedenstellend auszuüben. Er muss zu dieser Aktivität angewiesen werden, sei es über einen Befehl oder einen Knopfdruck. Um die Robustheitsanforderungen zu erfüllen, muss eine Robotik-Anwendung dabei Arbeitsanweisungen zielsicher und effizient entgegen nehmen. Erhält ein Roboter eine Aufgabe und führt stattdessen eine Andere aus, sei sie noch so perfekt gelöst, so ist die Anforderung nicht erfüllt.

In der Betrachtung der Mensch-Roboter-Interaktion geht es nicht alleine darum aufzuzeigen woraus die Interaktion besteht. Es sollen die Bedingungen für die Interaktion verstanden, eine Schnittstelle geschaffen und ihre Benutzbarkeit sichergestellt werden. Im Mittelpunkt steht also die Kommunikation zwischen Mensch und Roboter, ohne den Kommunikationskanal genauer zu spezifizieren.

Die Aufgabe der Mensch-Roboter-Interaktion ist es, komplexe Interaktionen zu vereinfachen. Die angebotene Benutzerschnittstelle soll dabei so einfach wie nur möglich gestaltet werden und dabei trotzdem alle notwendige Funktionalität bieten.

Schultz [GS07] definiert fünf Eigenschaften, welche die Interaktion zwischen Mensch und Roboter maßgeblich beeinflussen; Der Grad und das Verhalten der Autonomie, die Art des Nachrichtenaustausches, die Struktur des Teams, die Lernfähigkeit von Mensch und Roboter und die Gestalt der Aufgabe.

Status der HRI Bisher gibt es keine gute Idee für die ideale Benutzerschnittstelle eines Roboters. Viele verschiedene Schnittstellen wurden unabhängig voneinander entwickelt und evaluiert. Dabei haben sich jedoch noch keine Metriken etabliert.
Im Allgemeinen sind Ergebnisse in der Mensch-Roboter-Interaktion schwierig zu erzielen, da Ergebnisse nur über empirische Studien ermittelt werden können. Dies ist in der noch recht jungen Disziplin ohne bestehende Standardverfahren oder Bewertungsverfahren sehr Kosten und Zeit intensiv.
Hinzu kommen die grossen Unterschiede zwischen verschiedenen Robotikgebieten und Robotern. So sind zum Beispiel Versuche unter gleichbleibenden Testbedingungen besonders in der probabilistischen Robotik schwierig durchzuführen.

Langzeitziele der HRI Die Ziele in der Mensch-Roboter-Interaktion zentrieren sich um die Entwicklung von Qualitätskriterien. So müssen gute Metriken für Robotik-Anwendungen und allgemein einsetzbare Methoden für die Evaluation erarbeitet werden. Nur so können Prototypen entwickelt, evaluiert und anschließend verbessert werden. Dies kann dazu führen, dass Richtlinien erstellt und Standards definiert werden.

Um das Forschungsgebiet der Mensch-Roboter-Interaktion zu vertiefen, werden im folgenden die wichtigsten Definitionen und Begriffe erläutert.

Definition einer Aufgabe Eine Aufgabe kann jede mögliche Aktivität sein, die ein Anwender in einer Umgebung mit Hilfe einer Benutzerschnittstelle lösen soll. Eine Aufgabe unterscheidet sich von dem Konzept einer Aktion. Eine Reihe an Aktionen in einer virtuellen oder entfernten Umgebung können, müssen aber nicht bei der Performanz einer Aufgabe beitragen.
Definitert man eine Aufgabe in Bezug auf die Teleoperation, so besteht nach Parasuraman et al. [PSW00] eine Aufgabe aus vier Teilbereichen.

1. Sammeln von Informationen über den Roboter und seine Umgebung.
2. Analyse der gesammelten Informationen und Verstehen ihrer Bedeutung.
3. Entscheidungsprozess und Wahl der nächsten Aktion in der Bedienoberfläche.
4. Ausführen der nächsten Aktion.

Miller und Parasuraman [Mil07] kategorisieren Aufgaben in sogenannte Teilaufgaben und ordnen sie in einer Hierarchie ein. Ziel ist die Verbesserung der Performanz bei gleichbleibender Belastung des Anwenders.

Benutzerrollen In der HRI existieren unterschiedliche Rollen des Anwenders und damit unterschiedliche Informationsbedürfnisse. In der Mensch-Computer-Interaktion gibt es diese Rollen nicht. Man könnte zwar für den Entwickler oder Administrator unterschiedliche Rollen einführen, jedoch wird für gewöhnlich ausschließlich von einem Anwender ausgegangen.

Der Überwacher wird in Szenarien eingesetzt, bei denen mehrere Roboter zur gleichen Zeit im Einsatz sind. Er soll den Zustand aller Roboter kennen. Je nach Anwendung besitzt er die Möglichkeit einzugreifen oder den Operator zu informieren. Er benötigt dazu eine Schnittstelle, die ihm einen Überblick über die Situation verschafft und den Status aller Roboter auf einem Bildschirm visualisiert. Er muss mit Hilfe dieser Bedienoberfläche auftretende Probleme identifizieren oder Gefahrensituationen erkennen. Die Bedienoberfläche muss die Basis für Entscheidungen liefern, die Beziehung unter den Robotern darstellen und die Überwachnung von einer möglichst großen Anzahl von Robotern erlauben ohne den Anwender zu überlasten.

Der Operator interagiert direkt mit dem Roboter. Die Art der Steuerung hängt von der Autonomie des Roboters ab. Sie kann zwischen vollständiger Teleoperation bis hin zu komplexen Aufgabenbeschreibungen variieren. Befinden sich Roboter und Anwender nicht an einem gemeinsamen Ort, so soll sich der Operator durch den Roboter in der entfernten Umgebung präsent fühlen. Der Roboter stellt für den Operator ein Werkzeug dar. Weder das Werkzeug noch die Umgebung soll bei der Benutzung Schaden nehmen. Der Anwender muss deshalb ein Situationsbewusstsein entwickeln und pflegen. Die Herausforderung ist die Unterstützung dieses Bewusstseins trotz aller Einschränkungen durch Kommunikationsweg, Sichtfeld oder sonstiger eingeschränkter Wahrnehmung. Auch das Wiedererlangen des Situationsbewusstseins bei kurzzeitiger Ablenkung ist ein wichtiger Faktor.

Der Teamkollege ist eine Rolle, bei der die Master-Slave-Beziehung zwischen Roboter und Mensch vollständig aufgehoben ist. Mensch und Roboter bilden ein Team. Sie können untereinander Informationen austauschen und ein gemeinsames Ziel verfolgen oder sich gegenseitig unterstützen. Die beiden Mitglieder müssen dabei eine Vorstellung besitzen, welche Fähigkeiten der jeweilige Partner besitzt. Nur so kann eine effiziente Zusammenarbeit gewährleistet werden. Für gewöhnlich findet die Kommunikation über Sprache oder Gestik statt.

Der Entwickler wird eingesetzt, wenn ein Operator ein Problem nicht lösen kann. Er dient dazu schwerwiegende Probleme zu beseitigen, Reparaturen vorzunehmen und Fehlverhalten zu identifizieren.

Der Beisteher interagiert nicht aktiv mit dem Roboter. Er hält sich in der gleichen Umgebung, wie der Roboter auf. Der Roboter soll dabei die Anwesenheit eines Beistehers bemerken und situationsbezogen reagieren. Handelt es sich um einen Feind, so soll er sich vor ihm schützen. Auch für die Rolle des Beistehers ist interessant, ob dieser die Fähigkeiten des Roboters einschätzen kann.

Ein Anwender kann mehrere dieser Rollen gleichzeitig einnehmen, ausgeschlossen davon ist der Beisteher. Aber nicht alle Systeme haben oder brauchen alle Rollen.

2.3.1 Arbeitsbelastung

Die Arbeitsbelastung ist ein wichtiges Thema in der Mensch-Roboter-Interaktion. Wird ein Anwender zu sehr belastet, so sinkt die Aufmerksamkeit und die Wahrscheinlichkeit, dass Fehler bei der Bedienung auftreten, steigt.

Folgende Faktoren nehmen auf die Arbeitsbelastung direkten Einfluss:

Autonomie des Roboters Desto weniger autonom der Roboter agiert, umso mehr Arbeit hat der Anwender zu leisten.

Anzahl der gesteuerten Roboter Arbeiten Roboter nicht kollaborativ, sondern erwarten ausschließlich Arbeitsanweisungen des Anwenders, so wächst die Arbeitsbelastung mit der Anzahl der eingesetzten Roboter.

Komplexität der Schnittstelle Umso mehr unterschiedliche Informationen durch den Anwender verarbeitet werden müssen, desto schneller ist der Anwender kognitiv, also in seiner Wahrnehmung, dem Lernen, dem Erinnern und dem Denken, überlastet.

Komplexität der Welt Mit wachsender Komplexität der Welt wächst die Zahl der Gefahren, die ein Anwender berücksichtigen muss. Dies wirkt sich unmittelbar auf die Arbeitsbelastung aus.

Eine optimale Verarbeitung ist deshalb wichtig, um die Belastung des Anwenders möglichst gering zu halten. Eine gleichmäßige Verteilung von Arbeit vermeidet außerdem Flaschenhälse[1].

2.3.2 Situationsbewusstsein

Ein weiterer wichtiger Faktor in der Mensch-Roboter-Interaktion ist das Situationsbewusstsein. Situationsbewusstsein definiert, wie viel Wissen über den Zustand einer entfernten Umgebung vorhanden ist. Dabei ist sowohl das Situationsbewusstsein des Anwenders als auch eines autonomen Roboters selbst gemeint.
Man unterscheidet zwischen drei unterschiedlichen Graden des Situationsbewusstseins in einer Umgebung

Perzeption Der Anwender muss in der Lage sein wichtige Informationen über die Umgebung zu erhalten.

Verständnis Diese Informationen müssen verstanden, gespeichert und/oder weiterverarbeitet werden.

Projektion Einschätzungen über bevorstehende Ereignisse müssen möglich sein.

Auch können andere Einflüsse das Situationsbewusstsein beeinträchtigen oder verstärken. Der Grad der Arbeitsbelastung ist auch ausschlaggebend auf das aktuelle Situationsbewusstsein. Ist der Anwender zu sehr mit einem speziellen Problem beschäftigt, so wird er weitere Ereignisse in der Umgebung ausblenden.
Gleiches gilt für die Situation in der der Anwender mehr als einen Roboter steuert. Die einzelnen Roboter werden nur noch eine geteilte Aufmerksamkeit erfahren. Umso mehr Informationen der Anwender zu kontrollieren hat, desto eher wird er unter steigender Belastung Informationen übersehen. Systembedingte oder umweltbedingte Faktoren können ebenfalls das Situationsbewusstsein beeinträchtigen.

Situationsbewusstsein wirkt sich unmittelbar auf andere Konzepte der Mensch-Roboter-Interaktion aus, wie Vernachlässigung oder Interaktionszeit. Die Vernachlässigung ist ein Maß für die Aufmerksamkeit, die ein Roboter von einem Anwender erwartet. Ist ein Roboter weitgehend autonom, so kann er vernachlässigt werden, während er seine Aufgabe erledigt. Aber auch Zeitverzögerungen oder die Überlastung des Anwenders können dazu führen, dass die Performanz einbricht.

[1] langsamste Stelle, die zu geringerer Gesamtperformanz führt.

2.3.3 Gemeinsamkeiten

Für eine effektive Kommunikation zwischen Roboter und Anwender sind Gemeinsamkeiten vorteilhaft. Menschen entwickeln automatisch ein mentales Modell des Roboters und versuchen damit dessen Wissen abzuschätzen. Dieses Modell beeinflusst die Art, wie Anwender mit dem Roboter interagieren. Der Anwender wird einem schlauen Roboter weniger detailliert eine Aufgabe schildern. Sie teilen also eine gemeinsame Grundlage des Verständnisses.

Einer Großzahl heutiger Dialogsysteme mangelt es an Gemeinsamkeiten mit seinem Nutzer. Dialogsysteme werden häufig für Telefon-Hotlines eingesetzt. Macht der Anwender eine fehlerhafte Eingabe, so erhält er häufig eine Aufzählung aller möglichen Funktionen. Schon ein minimales Verständnis der Bedürfnisse des Anwenders würden diese Situation verbessern.

Wäre die Kommunikation mit einem Roboter weniger schwer, wie es bis heute der Fall ist, so wäre die Interaktion mit dem Roboter sofort zufriedenstellender. Es sollte nicht notwendig sein die Fähigkeiten eines Roboters aufzählen zu müssen, bevor man mit ihm interagieren kann. Es sollte nahe liegen, welche Funktionalität ein Roboter unterstützt und welche nicht. Hierbei ist das Aussehen des Roboters entscheidend für eine Ersteinschätzung. Umso menschlicher der Roboter aussieht, desto besser ist er einzuschätzen oder mit einem Stereotyp zu verbinden. Die Interaktion mit einem Roboter, der wie ein Kind aussieht, im Gegensatz zu einem Roboter mit dem Aussehen eines Türstehers, würde vollkommen unterschiedlich beginnen.

Umso mehr ein Roboter sich den Fähigkeiten eines Menschens anpasst, desto weniger muss sich der Mensch für den Roboter anpassen.

2.4 Evaluation

Aus der Mensch-Computer-Interaktion (HCI) sind eine Vielzahl von Evaluationsmethoden für die Softwareentwicklung hervorgegangen. Pionierarbeit auf diesem Gebiet haben Card et al. [CMN83] mit ihrem Buch „The Psychology of Human Computer Interaction“ geleistet. Sie beschreiben einen analytischen Weg zur quantitativen Evaluation von Benutzeroberflächen genannt GOMS.
Drury et al. [DSK07] liefert eine Studie zur Adaption von GOMS für die Mensch-Roboter-Interaktion. Für die Mensch-Roboter-Interaktion ist nicht nur die Benutzbarkeit einer Anwendung von Interesse, sondern auch das Situationsbewusstsein und die gemeinsame Grundlage zwischen Roboter und Anwender.

2.4.1 Metriken

Metriken stellen einen Weg Beurteilung von Qualität dar. Sie beurteilen wie gut oder schlecht gewisse Kriterien erfüllt werden oder welche Eigenschaften eine Anwendung hat.

Die Schwierigkeit bei der Mensch-Roboter-Interaktion ist Vergleichbarkeit. Es müssen Maßeinheiten gefunden werden, mit welchen sich möglichst generische Testaufbauten vergleichen lassen.
Viele Metriken beruhen auf Ansätzen, wie einer Zeit bis zur Fertigstellung. Diese Metrik ist stark aufgabenabhängig und lässt den Vergleich von unterschiedlichen Roboter-Anwendungen nicht zu. Auch würde ein perfekt bedienbarer und völlig autonomer Roboter gegen einen schnellen kontinuierlich gesteuerten Roboter verlieren.

Verschafft man sich einen Überblick über die bisher entworfenen Metriken, so findet man eine Vielzahl von Metriken, die sich auf spezielle Gebiete der Robotik beziehen.

Themengebiet	Metrik
Bergungsroboter	Zeit, die für definierte Aktivitäten benötigt wurde. Wie umfassend das Gebiet erkundet wurde. Ausgang von kritischen Vorfällen.
Straßenfahrzeuge	Simulation Höhe der Arbeitsbelastung.
Geländefahrzeuge	Feldstudien Zeit, die für Eingriffe des Operators benötigt wurde. Zeit zur Erlangung von Situationsbewusstsein. Höhe der Arbeitsbelastung.

Herausgebildet haben sich ebenfalls gebietsunabhängige Metriken, wie die *Vernachlässigungstoleranz*. Ein Maß für die Autonomie eines Roboters. Man ist damit in der Lage zu messen, wie stark die Performanz einbricht, wenn der Roboter eine Zeit lang vernachlässigt wird. Eine weitere Metrik ist die *Bedienungseffizienz*, sie beschreibt die Zeit, die benötigt wird Situationsbewusstsein zu erlangen, einen Plan zu erstellen und diesen dem Roboter mitzuteilen. Die *Toleranz der Weltkomplexität* ist eine Metrik, die die Skalierbarkeit eines Interaktionsschemas misst.
Das Gegenstück der Vernachlässigungstoleranz ist die *Notwendige Aufmerksamkeit*, sie betrachtet die durchschnittliche Zeit, die zur Bedienung des Roboters notwendig ist im Vergleich zur Dauer der Vernachlässigung.

2.4.2 Studienarten

Zu den beiden wichtigsten Studienarten gehören die Feld- und Laborstudien.
Feldstudien eigenen sich nur zur Evaluation bestehender Pratiken. Die Einführung neuer Technologie wird diese ändern.
Laborstudien sind vorteilhaft für die Isolation von Variablen, um Effekte festzustellen. Außerdem lassen sich frühe Prototypen testen. Schwierig ist die Simulation der Kondition in der realen Welt.

2.5 Android

Am 5. November 2007 veröffentlichte die Open Handset Alliance das erste Mobiltelefon mit Android-Betriebsystem [CBS]. Es handelt sich um ein Konsortium von unterschiedlichen Firmen aus dem Umfeld der Mobilfunkbranche, welche das gemeinsame Ziel von offenen Standards für mobile Geräte verfolgen. In der Vergangenheit war das Betriebssystemangebot für mobile Endgeräte stark fragmentiert. Die am stärksten verbreiteten Betriebssysteme waren Windows CE, Symbian und Palm OS. Für Entwickler war es zu diesem Zeitpunkt unmöglich Software für Endgeräte unterschiedlicher Hersteller anzubieten. Schon die Softwareentwicklung für Endgeräte eines einzelnen Herstellers gestaltete sich auf Grund unterschiedlicher Merkmale, wie Bildschirmtyp, Bildschirmgröße, Tastenlayout oder fehlender Spezifikationen als schwierig.
Den ersten Versuch eine gemeinsame Plattform zu schaffen hat SUN Microsystems mit seinem Produkt Java Micro Edition unternommen. Dabei handelt es sich um eine schlanke Java-Plattform speziell entwickelt für die beschränkten Ressourcen mobiler Geräte. Entwickler erhielten zum ersten Mal die Möglichkeit Software unabhängig vom Endgerät zu entwickeln. Durch das Angebot von proprietären Erweiterungen hat die Standardisierung schnell wieder Einbußen erfahren.

Google hat durch das Angebot des Android-Betriebssystems die offenen Standards auf eine neue Stufe gehoben. Android wird unter einer freien Software-Lizenz angeboten und erfreut sich zunehmender Beliebtheit. Nicht nur Geräteherstellern kommt das kostengünstige Betriebssystem zugute, auch unter Anwendern und Entwicklern genießt es eine stetig wachsende Akzeptanz.
Android liefert eine standardisierte Plattform zur einfachen Entwicklung neuer Anwendungen und der Garantie einer geräteübergreifenden Lauffähigkeit. Mit Hilfe eines Kompatibilitätsprogramms wird dafür Sorge getragen, dass keine inkompatible Android-Implementation vertrieben wird. Besteht eine Implementation den Kompatibilitätstest, so darf sie den Markennamen Android™ tragen und samt *Google Play* (früher unter den Namen Android Market bekannt), einem Marktplatz zur einfachen Installation zusätzlicher Anwendungen, ausgeliefert werden.

Die Hardware-Anforderungen des Betriebssystems sind äußerst gering. Begonnen wurde die Entwicklung auf Basis von stromsparenden Prozessoren der Firma ARM mit mindestens 200 MHz. Die leichte Portierbarkeit sorgte schnell für weitere unterstützte Architekturen und ein breit gefächertes Einsatzfeld weit über Smartphones hinaus. Eine weitere Hardware-Anforderung ist der Einsatz eines Touchscreens. Darüber hinaus waren zu Beginn vier physikalische Tasten spezifiziert, welche die Aufgabe von Standard-Bedienelementen einnehmen sollten. Besitzt ein Entwickler keine Verwendung für die Menü-Taste, so zeigt das Android-Gerät bei der Betätigung keine Reaktion. Auf Grund dieses Mangels an Benutzerfreundlichkeit trennte man sich von diesen permanenten Tasten. Aktuelle Geräte besitzen diese Tasten nicht mehr. Dies kommt insbesondere Android-Tablets zu Gute. Seitdem werden jene Bedienelemente, wenn der Bedarf

besteht, grafisch dargestellt. Eine Kompatibilität zu älterer Software ist so gewährleistet. In den weiteren Komponenten ist das System äußerst flexibel und liefert kaum Einschränkungen. Es gibt keine festen Vorgaben für eine Bildschirmgröße, der Verwendung eines bestimmten Eingabegerätes oder der angebotenen Kommunikationswege.
Die Anwendungsentwicklung für diese Plattform erfolgt in erster Linie in der Programmiersprache Java. Mit Hilfe des von Google angebotenen Android Software Development Kits (Android SDK) wird auf Wunsch der Programmcode, Daten und Ressourcen zu einem Installationspaket zusammen geschnürt. Das Ergebnis ist ein komprimiertes Archiv mit signiertem Inhalt. Die Datei trägt die Endung *APK* und kann auf einem Android-Gerät ohne weitere Maßnahmen installiert werden. Um Sicherheit zu gewährleisten werden Anwendungen in einem sogenannten Sandkasten ausgeführt. Anwendungen erhalten dabei nur so viele Rechte, wie zum Betrieb notwendig sind.

Im folgenden wird der Systemaufbau beschrieben, um einen Überblick über die einzelnen Schichten und der zur Verfügung gestellten Komponenten zu geben.

2.5.1 Systemaufbau

Linux-Kernel Ein Linux-Kernel bildet das Herzstück von Android. Er basiert auf Version 2.6 und stellt die Kernfunktionalität des Systems zur Verfügung. Dazu gehören ein besonderes Speichermanagement, die Verwaltung von Prozessen, der Verteilung von Rechenleistung, sowie einem Sicherheits- und Treibermodell.
Er bildet die Abstraktionsschicht zwischen der Hard- und Software. Für einen Hersteller ist dies die einzige Schicht, die angepasst werden muss, um ein neues Gerät zu integrieren. Alle weiteren Veränderungen auf anderen Schichten, die gerne vorgenommen werden, sind kosmetischer Natur. Sie dienen zur Repräsentation einer Marke, einer herstellerspezifischen Anmutung und der Abgrenzung von der Konkurrenz. Die Notwendigkeit einer Anpassung besteht jedoch nur in der Abstraktion der Hardware.

Laufzeitumgebung Die Laufzeitumgebung für Android-Anwendungen bildet die Dalvik Virtual Machine (DVM). Es handelt sich um eine registerbasierte virtuelle Maschine. Die Besonderheit ist das Ausnutzen der Tatsache, dass der im Endgerät eingesetzte Prozessor ebenfalls Register besitzt. Klassische JVMs (Java Virtual Machines) nutzen ein allgemeineres Modell eines Prozessors und müssen deshalb Performanzeinbußen in Kauf nehmen. Die DVM basiert auf der quelloffenen JVM *Apache Harmony* und wurde speziell für den Einsatz auf mobilen Geräten optimiert. Sie benötigt deshalb nur einen minimalen Speicherbedarf pro Instanz. Ausgeführt wird von ihr das „Dalvik Executable Format“ (DEX). Jede Android-Anwendung wird in einem eigenen Systemprozess gestartet, in dem eine DVM läuft. In dieser virtuellen Maschine wird die eigentliche Anwendung gestartet. Dies nimmt zwar mehr Ressourcen in Anspruch, als wenn alle Anwendungen in einer gemeinsamen

virtuellen Maschine gestartet werden, jedoch bietet es Vorteile in Sicherheit und Verfügbarkeit. Durch die Vermeidung gemeinsamer Speicherbereiche wird gewährleistet, dass ein abstürzender Prozess keine weiteren Anwendungen mit sich zieht. Der Einsatz unterschiedlicher Prozesse unter Verwendung einer separaten Benutzerkennung pro Anwendung sorgt für Sicherheit. Der Webbrowser Google Chrome nutzt ein vergleichbares prozessbasiertes Konzept, so dass der Absturz eines einzelnen Tabs nicht den gesamten Browser beeinträchtigt. Während der Entwicklung von Android-Anwendungen wird ein herkömmlicher Java-Compiler eingesetzt. Er sorgt für die Übersetzung des Programmcodes zu Java-Bytecode. Dies hat den Vorteil, dass die Entwicklung in einer bereits etablierten Sprache und einer bekannten Entwicklungsumgebung erfolgen kann. Ein vom Android SDK zur Verfügung gestellter Cross-Compiler namens *dx* sorgt für die Konvertierung des Java-Bytecodes in das DVM-kompatible Format DEX. Über diesen Weg vermeidet Google an SUN Lizenzkosten entrichten zu müssen. Es wird weder eine geschützte JVM auf dem Endgerät eingesetzt, noch Java-Bytecode ausgeführt. Java ist ausschließlich ein Zwischenprodukt, welches während der Entwicklung entsteht.

Bibliotheken Um dem Android-System Funktionalität zu verleihen kommen eine Vielzahl von systemnahen Bibliotheken zum Einsatz. Sie werden für zeitkritische Aufgaben eingesetzt, wie das Abspielen und Aufzeichnen von populären Audio- und Video-Formaten, dem hardwarebeschleunigten Rendern von zwei- und dreidimensionalen Bildern oder der Darstellung von Schriftarten. Einen standardisierten Zugriff auf diese Eigenschaften erfolgt mit Hilfe des Anwendungsrahmen. Mit Hilfe des Java Native Interfaces (JNI) erhält der Entwickler darüber hinaus die Möglichkeit weitere Bibliotheken als native Methoden in der virtuellen Maschine verfügbar zu machen. So ist es möglich die Funktionalität einer in C implementierten Anwendung, wie OpenCV innerhalb einer in Java geschriebenen Android-Anwendung zu nutzen.

Anwendungsrahmen Der Anwendungsrahmen stellt die Funktionalität zur Verfügung, auf welche der Programmierer bei der Entwicklung einer Anwendung zurückgreift. Diese Schicht wird auch als Android-Schnittstelle (Android API) bezeichnet. Sie ist vollständig in Java geschrieben und stellt Systemklassen für den Zugriff auf die Hardwarekomponenten und Bibliotheken bereit. Diese Schnittstelle ist gut dokumentiert und mit vielen Code-Beispielen illustriert. Mit der Entwicklung des gesamten Android-Systems hat auch die Schnittstelle eine kontinuierliche Veränderung erfahren. Sie ist aus diesem Grund mit Hilfe eines ganzzahligen Wertes versioniert. Diese Versionsnummer wird von Google als „API Level“ bezeichnet. Derzeit befindet sich die zugehörige API zu Android 4.0.3 bei Level 15. Die gesamte Funktionalität, die eine Android-Version bietet ist über diese Schnittstelle abgebildet. Identifizierbar sind Systemklassen über den Suffix „Manager“.

2.5.2 Entwicklung von Anwendungen

Das von Google zur Verfügung gestellte Android SDK liefert alle notwendigen Werkzeuge für die Erstellung und das Testen einer Android-Anwendung.
Es setzt sich zusammen aus Hilfsprogrammen, versions-spezifischer SDKs und deren Erweiterungen. Ein Hauptbestandteil ist der „Android SDK and AVD Manager“. Über dieses Programm werden die Komponenten des SDKs verwaltet, installiert und aktualisiert. Eine weitere Funktionalität ist die Erzeugung und Verwaltung von virtuellen Android-Geräten unterschiedlicher Versionen und Bildschirmgrößen.
Mit Hilfe des Android Emulators können diese virtuellen Geräte gestartet werden. Emuliert wird dabei ein vollständiges Gerät samt ARM-Prozessor, so dass ein konventionelles Android-System gebootet werden kann. So wird erreicht, dass sich das Gerät gleichsam wie ein physikalisches Android-Gerät verhält. Im Gegensatz zum Apple iOS SDK handelt es sich nicht um eine Simulation der eigentlichen Anwendung. Simuliert werden lediglich einzelne Hardware-Komponenten, wie der GPS-Empfänger oder die Mobilfunkeinheit. Ziel ist die Abbildung des gesamten Funktionsumfangs eines echten Gerätes, um für den Programmablauf wichtige Ereignisse nachstellen zu können.

Bis heute weist die Umsetzung jedoch noch Lücken auf. Sowohl eine üblicherweise verfügbare Kamera als auch der Einsatz von Multitouch-Gesten werden vom Emulator bisher nicht unterstützt. Mit Hilfe des Mauszeigers des Host-Systems lässt sich ausschließlich die Interaktion eines einzelnen Fingers abbilden. Der Einsatz von virtuellen Fingern oder Tastenkombinationen ist nicht möglich. Eine weitere Schwierigkeit betrifft die Netzwerkkommunikation. Latenzen und eine beschränkte Bandbreite lassen sich zwar definieren, jedoch werden die emulierten Geräte von einem virtuellen Router maskiert. Ein direkter Zugriff auf beliebige Ports des Übertragungssteuerungsprotokolls (TCP) ist deshalb nicht möglich. Lediglich die Weiterleitung einzelner ausgewählter Ports ist mittels der *Android Debug Bridge* (ADB) konfigurierbar.

Die Android Debug Bridge ist ein vielseitig eingesetztes Hilfsmittel bei der Entwicklung von Android-Anwendungen. Für den Einsatz der ADB muss die Funktion „USB Debugging“ in den Geräte-Einstellungen aktiviert sein. Man wählt dazu im Abschnitt „Apps“ den Unterpunkt „USB-Debugging“ und setzt einen Haken. Nach der erfolgreichen Verbindung mit einem Endgerät über USB oder Netzwerk ist es unter anderem möglich Dateien zu lesen oder zu schreiben, einen kontinuierlichen Logbuchauszug anzufordern oder Anwendungen in Form von APK-Dateien zu installieren und zu löschen. Dieser Weg wird automatisiert genutzt, um während der Entwicklung Testversionen auf das Gerät zu übertragen oder den Einblick in einen Stack-Trace zu erhalten, um die Ursache für einen Programmabsturz festzustellen.

Entwicklungsumgebung Als eine integrierte Entwicklungsumgebung (IDE) für Android-Anwendungen wird Eclipse [Foua] empfohlen. Es handelt sich um eine populäre und hauptsächlich für die Java-Entwicklung konzipierte IDE. Sie ist freie Software und ist über Plugins, um neue Funktionalität erweiterbar. Diesen Weg hat Google gewählt und stellt den Plugin „Android Developer Tools“ zur Verfügung. Dieser Plugin ermöglicht das einfache Einbinden des Android SDKs in Eclipse. Dabei hilft er bei der Erstellung neuer Android-Projekte, dem direkten Kompilieren und Ausführen von Anwendungen und durch eine umfassende Tooltip-Unterstützung für die angebotenen Klassen und Methoden der gewählten SDK-Version. Er liefert darüber hinaus zusätzliche Ansichten zum Profiling und der Fehlersuche während der Laufzeit einer Anwendung mittels dem Dalvik Debug Monitor Server (DDMS). Auch ein visueller Layout-Editor ist seit kurzem ein fester Bestandteil und erlaubt die einfache Erstellung von Benutzeroberflächen.

Wird ein kommandozeilen basierter Mechanismus zum Bau der Software bevorzugt, zum Beispiel beim Einsatz von GNU Emacs [Foub] als Entwicklungsumgebung, so genügt die Installation von *Apache Ant*. Mit Hilfe des Android Managers kann auf der Kommandozeile eine Projektstruktur erzeugt werden. Sie enthält die Datei `build.xml`, welche alle notwendigen Informationen zur Verfügung stellt, um mit Hilfte von `ant` ein Projekt zu kompilieren und auf einem Endgerät mittels ADB zu installieren.

Projektstruktur Android-Projekte besitzen eine vorgeschriebene Verzeichnisstruktur die vom Build-Mechanismus während des Kompiliervorgangs durchsucht wird. Das Wurzelverzeichnis eines jeden Projektes besteht aus den folgenden Komponenten:

AndroidManifest.xml Die Datei enthält eine Beschreibung der Anwendung und wird unverändert in die Android-Paketdatei übernommen. Die Android-Plattform nutzt diese Informationen um die Anwendung im System zu integrieren. Zu den obligatorischen Feldern gehört der Programmname, ein eindeutiger Paketname, die mindestens notwendige Android-Version und die *Aktivität*, die beim Start der Anwendung ausgeführt werden soll. Benötigt ein Programm für den Betrieb die Erlaubnis auf geschützte Hard- oder Software, werden diese ebenfalls im Manifest eingetragen und bei der Installation der Anwendung angefragt. Darüber hinaus können so genannte *Absichten* definiert werden, auf welche die Anwendung reagieren soll und *Dienste*, um die das System erweitert wird.

build.xml Die Build.xml ist das Regelwerk des Build-Mechanismus. Sie ist vergleichbar mit einem `Makefile` und enthält so genannte Aufgaben, mit der die Anwendung kompiliert und zu einer signierten Android-Paketdatei zusammengefasst wird. Die verschiedenen Aufgaben werden von `ant` über Kommandozeilenparameter angeboten. Der Aufruf von `ant installr` kompiliert und

installiert zum Beispiel ein Veröffentlichungsversion der Anwendung auf ein per Android Debug Bridge angebundenes Gerät.

bin/ Die fertig kompilierte Anwendung und unterschiedliche Zwischenerzeugnisse, wie ein Bündel der Klassen im Dalvik Executable Format werden in diesem Verzeichnis abgelegt.

libs/ Dieses Verzeichnis kann Bibliotheken von Dritten im Java-Archiv Format (JAR) enthalten, welche zur Laufzeit der Anwendung benötigt werden. Sie werden in einem Paket mit der Anwendung ausgeliefert. Der Build-Mechanismus sorgt dafür, dass der enthaltene Java-Bytecode in ein DVM-kompatibles Format konvertiert wird.

res/ enthält Ressourcen wie XML-Dateien oder Grafiken. Über XML-Dateien werden entweder Bildschirmlayouts beschrieben oder Konstanten definiert. Android bietet zwei Wege um eine Benutzeroberfläche zu generieren. Entweder werden die Elemente mit Hilfe einer XML-Datei deklariert oder direkt im Programmcode instanziiert. Ersteres bieten den Vorteil einer klaren Trennung zwischen Oberfläche und der Programmlogik. Die Unterstützung dieser XML-basierten Layouts vereinfacht außerdem den Einsatz von GUI-Designern. Insbesondere beim Wiedereinlesen von Benutzeroberflächen-Beschreibungen eignet sich XML besser als erzeugter Programmcode. In fest definierten Unterverzeichnissen werden individuelle Grafiken und Icons für Bildschirme mit unterschiedlicher Punktdichte abgelegt. Hierfür wurde die Punktdichte von Bildschirmen in vier unterschiedliche Klassen eingeteilt: niedrig, durchschnittlich, hoch und sehr-hoch. Die gesamten Ressourcen werden komprimiert und sind Quelltext über eindeutige Bezeichner direkt und schnell referenzierbar.

assets/ sind mit Ressourcen vergleichbar, jedoch gibt es hier keine Einschränkungen bezüglich des Dateityps. Außerdem werden die Inhalte dieses Ordners nicht komprimiert. Anders als auf Ressourcen gibt es zum Zugriff auf die Dateien keine direkten Referenzen. Der Zugriff auf die Rohdaten erfolgt ähnlich, wie in einem Dateisystem.

src/ Dieses Verzeichnis enthält den gesamten Quelltext der Anwendung. Die darin enthaltene Verzeichnisstruktur spiegelt den verwendeten Paketnamen wieder. Ein unter Java-Entwicklern verbreiteter Standard und ebenfalls vom Build-Mechanismus erwartete Hierarchie.

gen/ enthält automatisch erzeugten Quelltext. Das Android Asset Packaging Tool (aapt) ist für die Generierung verantwortlich und liefert damit das Bindeglied zwischen Ressource und Quelltext. Wird Eclipse eingesetzt, so erfolgt der Aufruf von aapt nach jeder Änderung einer Ressource automatisch.

Aktivitäten Eine Anwendung auf einem Android-Gerät nimmt für gewöhnlich den gesamten Bildschirm ein. Fenster, wie man sie von einem Desktop kennt mit Fensterheber und Dekoration gibt es in diesem Sinne nicht. Eine solches Vollbild

wird als Aktivität bezeichnet. Die in einer Aktivität dargestellten Elemente, wie Texte, Bilder oder Knöpfe sind sogenannte Ansichten. Wechselt man von einem Bildschirm zum Nächsten, so findet ein Wechsel zwischen zwei Aktivitäten statt. Auch wenn es ein animierter Übergang zwischen zwei logisch zusammenhängenden Bildschirminhalten anders vermutlich lässt, so erfolgt dabei ein Aufruf einer weiteren Aktivität. Durch den Einsatz einer Stapelverarbeitung wird ein dynamischer Programmfluss ermöglicht. Illustriert ist ein solcher Stapel in Abbildung 1.

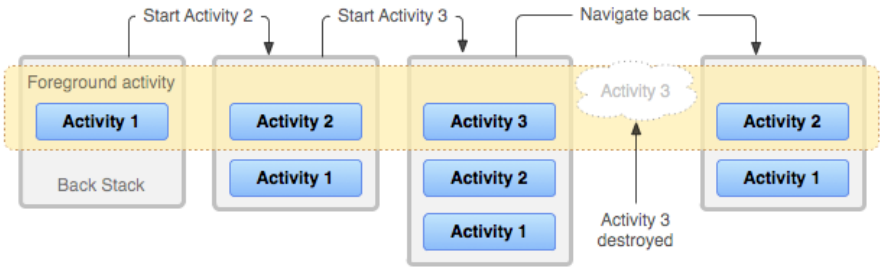


Abbildung 1: Stapelverarbeitung [Proa]

Er erlaubt das Zurückspringen zu vorangegangen Aktivitäten, um Fehler zu korrigieren oder eine unterbrochene Aktivität fortzusetzen. Der Ursprung des Stapels besteht immer aus dem Startbildschirm, der auch über die Home-Taste erreichbar ist. Nicht sichtbare Aktivitäten werden unweigerlich pausiert und erhalten keine Rechenzeit solange sie sich im Hintergrund befinden. Stellt das System einen Mangel an Arbeitsspeicher fest, so werden ausgewählte pausierte Aktivitäten gestoppt und zerstört. Aktivitäten sind sehr kurzlebig und deshalb ausschließlich für die Interaktion mit dem Benutzer vorbehalten. Aufwendige Berechnungen oder eine unter Umständen träge Kommunikation über eine Netzwerkverbindung sollte nicht innerhalb der Aktivität erfolgen. Jede unbedachte Belastung dieses Benutzerschnittstellen-Threads (UI-Threads), führt zu einer Verschlechterung der Ansprechbarkeit der grafischen Oberfläche. Weder Benutzeingaben sind dann mehr möglich, noch erfolgt eine Aktualisierung der Bedienelemente. Erkennt das Android-System einen für mehr als fünf Sekunden geblockten UI-Thread so wird dem Anwender ein „Programm reagiert nicht“-Dialog präsentiert und die Möglichkeit eröffnet die Anwendung zu beenden. Benutzer verbinden mit diesem Dialog häufig einen Programmabsturz, obwohl es in erster Linie auf eine unsaubere Programmierung hindeutet.

Lebenszyklus einer Aktivität Durch das spezielle Konzept einer Android-Anwendung aus lose verknüpften Aktivitäten entsteht der Bedarf eines besonderen Lebenszyklus. Jede Aktivität besitzt ihren eigenen Lebenszyklus. Dieser Lebenszyklus ist in Abbildung 2 schematisch dargestellt. Es existiert kein ganzheitlicher Zyklus für eine vollständige Anwendung oder der Instanz einer virtuellen Maschine. Wird eine Anwendung gestartet, so wird die im Manifest definierte Start-Aktivität aufgerufen. Die Aktivität wird erstellt (Create), gestartet (Start) und die Benutzeroberfläche dem Anwender angezeigt. Rückt eine Aktivität in den Hintergrund, so

wird sie pausiert. Dies geschieht zum Beispiel während des Einblendens eines Dialogs. Nach der Bearbeitung des Dialogs rückt die pausierte Aktivität wieder in den Vordergrund und läuft wieder (Resume). Wird eine nächste Aktivität gestartet und ist die Vorangegangene für den Anwender nicht mehr sichtbar, so wird sie gestoppt (Stop) und der Systemprozess mit der DVM-Instanz beendet. Sobald sich eine Aktivität im Zustand pausiert, gestoppt oder zerstört befindet, hat das Betriebsystem die Möglichkeit bei einem Mangel an Arbeitsspeicher diese vollständig zu beenden. Dabei wird das Ziel verfolgt Aktivitäten möglich lange im Arbeitsspeicher vorzuhalten, um eine maximale Reaktionsfähigkeit zu gewährleisten.

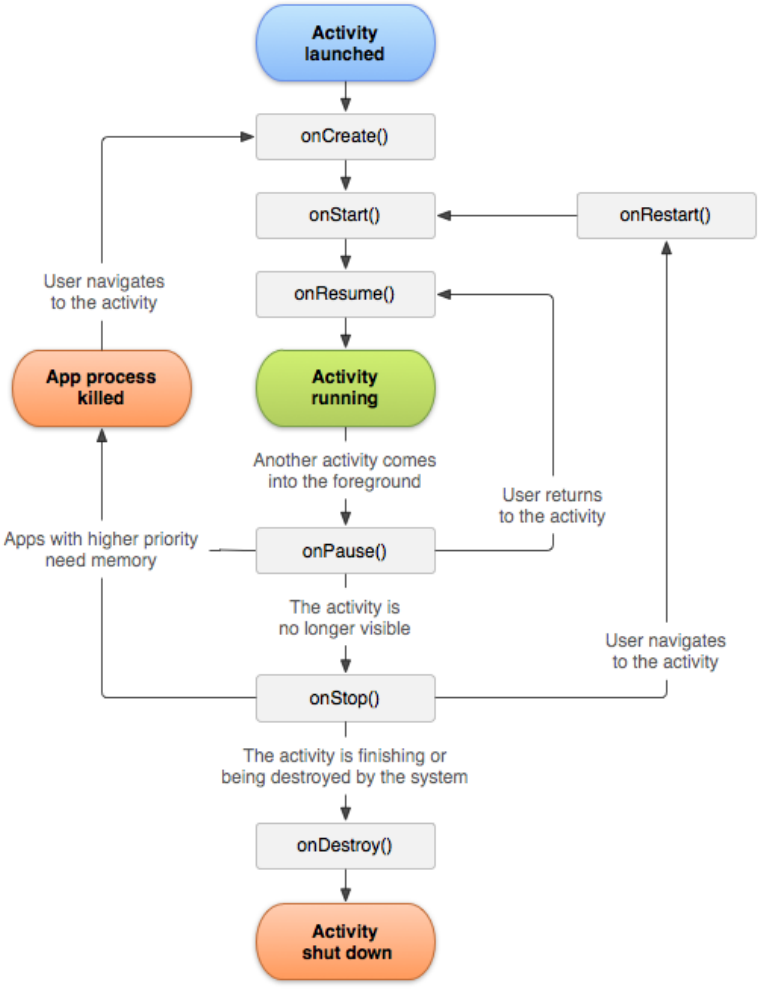


Abbildung 2: Lebenszyklus einer Aktivität [Prob]

Dienste Das Gegenstück zu Aktivitäten sind die Dienste. Sie besitzen keine grafische Oberfläche, sondern verrichten ihre Arbeit vollständig im Hintergrund. Dabei sind sie vergleichsweise langlebig und stellen ihre Arbeit erst ein, wenn sie aktiv angehalten werden. Ihr Start kann an Ereignisse geknüpft werden oder aus einer Aktivität heraus erfolgen. Sie ermöglichen Funktionen, wie das

periodische Aktualisieren von Neuigkeiten, einer persistenten Verbindung zu einem Sofort-Nachrichten-Dienst oder dem Abspielen von Musikstücken. Eine Anwendung zur Musikwiedergabe besteht folglich aus einer Aktivität, ähnlich einer Fernbedienung, die einen Dienst steuert. Wird eine andere Aktivität aufgerufen, so kann die Wiedergabe trotzdem fortgesetzt werden.

Für einfache Aufgaben, wie dem einmaligen Nachladen einer Datei beim ersten Programmstart wurde ab Android 1.5 eine Hilfsklasse eingeführt. Sie trägt den Namen *AsyncTask* und liefert ein einfaches Thread-Management, um ein kontrolliertes nebenläufiges Ausführen von Aufgaben zu erlauben ohne die Benutzeroberfläche zu blockieren.

Fragmente Mit Android 3.0 wurden sogenannte Fragmente eingeführt. Sie schaffen eine bessere Modularisierung von Aktivitäten und erlauben eine ausgefeiltere Gestaltung von Benutzeroberflächen für unterschiedliche Bildschirmgrößen, ohne dass tiefgreifende Änderungen im Programmcode vorgenommen werden müssen.

2.6 Roboter Operation System

Die Software-Basis des eingesetzten Roboters bildet das *Roboter Operation System* (ROS) [QGC+09]. Es handelt sich um ein Open-Source-Projekt des Robotikinstituts Willow Garage. Das Projekt erfreut sich starker Beliebtheit und einer großen Entwicklergemeinde. Geboren wurde ROS im Jahre 2007 an der Universität Stanford im Rahmen des Stanford AI Robot (STAIR) Projektes [QBN07]. Damals trug die Software den Namen „switchyard“. Die Dokumentation und das Angebot von Dritten wird auf einer großen Community-Webseite [Garb] gepflegt.

Hauptaufgabe von ROS ist die Entwicklung und Pflege einer Robotik-Anwendung zu unterstützen und einen Standard für die Kommunikation bereitzustellen.
Es ist lauffähig unter Linux oder Mac OS X. Bei ROS handelt sich nämlich nicht, um ein Betriebssystem im herkömmlichen Sinne, sondern um eine Grundstruktur, welche eine Sammlung an Bibliotheken, Hilfsprogrammen und Techniken zur Verfügung stellt. Die Software sollte deshalb als Vermittlungssoftware (Middleware) angesehen werden. Es finden sich jedoch viele Konzepte eines konventionellen Betriebssystems wieder, wie sie auf einem heterogenen Rechnerverbund eingesetzt werden. Beispiele sind eine Hardwareabstraktion, Gerätetreiber, Nachrichtenaustausch zwischen einzelnen Knoten und eine Paketverwaltung. Die Paketverwaltung prüft und gewährleistet Abhängigkeiten zwischen den einzelnen Komponenten. Die Komponenten werden als *Paket* bezeichnet. Eine Sammlung von mehreren Paketen nennt sich *Stack*.
Ein großer monolithischer Kern, wie er schnell bei Robotik-Anwendungen entsteht, sei es aus Gründen der Effizienz oder bedingt durch einen Wachstumsprozess, soll vermieden werden. Stattdessen wird ein Mikrokernel-Design eingesetzt, bei welchem eine große Anzahl kleiner Komponenten der Grundstruktur seine Funktionalität verleiht. Dabei hat die Kontrolle über Stabilität und Komplexität einen höheren Stellenwert als der Anspruch ein Problem effizient zu lösen.

Vermieden werden soll darüber hinaus die Gebundenheit an eine einzelne Programmiersprache. Jeder Entwickler besitzt seine persönlichen Vorzüge und Abneigungen aus unterschiedlichsten Beweggründen. ROS unterstützt deshalb eine Vielzahl von Programmiersprachen. Dies ist möglich, da alle Spezifikationen ausschließlich auf der Nachrichten-Ebene bestehen. Jegliche Kommunikation findet auf Basis von *XML-RPC* statt. Dabei handelt es sich ebenfalls um eine Spezifikation, bei dessen Entwurf darauf geachtet wurde, dass sie ohne großen Aufwand in unterschiedlichen Programmiersprachen zu implementieren ist. Die Hürde eine ROS-Implementation in einer weiteren Sprache zu entwickeln, ist somit so niedrig wie nur möglich.

Das Fundament einer ROS-Implementation besteht aus vier Komponenten: *Knoten*, *Nachrichten*, *Themen* und *Dienste*.

2.6.1 Knoten

Knoten sind Systemprozesse, die einzelne kleine Aufgaben einer Roboter-Anwendung erfüllen. Das Konzept hinter ROS empfielt, dass ein Knoten eine einzelne Aufgabe erfüllt. Das kann die Berechnung eines Algorithmus sein oder die Kommunikation mit einer Hardware-Komponente. Eine vollständige Anwendung besteht aus einer Vielzahl unabhängiger Knoten, die alle zur Gesamtfunktionalität beitragen. Diese Knoten müssen nicht notwendigerweise auf ein und dem selben System laufen, ebenso wenig müssen sie in einer gemeinsamen Programmiersprache geschrieben sein.
Die Kommunikation zwischen den Knoten erfolgt über einen direkten zuverlässigen bidirektionalen Datentransport (direkte TCP-Verbindung) auf peer-to-peer Basis. Zur Laufzeit findet somit eine lose Verknüpfung unter den Knoten statt. Ausgetauscht werden dabei *Nachrichten*.
Die Implementierung einer verbindungslosen und ungesicherten Transportschicht (UDP) befindet sich in Entwicklung. Sie kommt Anwendungen zu Gute, die auf eine geringe Latenz angewiesen sind und für die Zuverlässigkeit kein notwendiges Kriterium darstellt.

Ein spezieller Knoten namens *master* wird eingesetzt, um Knoten untereinander bekannt zu machen. Beim Start eines Knotens muss deshalb nur eine notwendige Bedingung erfüllt sein, nämlich die Erreichbarkeit eines Masters. Dies reicht aus, um die restlichen Knoten im System aufzufinden und eine Kommunikation aufzubauen. Der Master pflegt dafür Listen von bekannten Knoten und deren angebotenen Kommunikationswege.

Der Versand und Empfang von Nachrichten erfolgt durch die jeweilige ROS-Implementation der entsprechenden Programmiersprache. Dies macht den Austausch von Nachrichten sehr robust und eliminiert häufige Fehlerquellen. Eine in Knoten immer wiederkehrende Funktionalität ist leicht nutzbar.

2.6.2 Nachrichten

Mit Hilfe von Nachrichten lassen sich beliebige Information übertragen. Von einfachen Fließkommazahlen bis hin zu abstrakten Datentypen, wie den Punktwolken eines Tiefenbildes. Eine Nachricht ist eine typisierte Datenstrukturen, welche aus primitiven Datentypen zusammen gesetzt ist. Nachrichten dürfen bis zu einer beliebigen Tiefe ineinander verschachtelt werden.
Im folgenden ein Beispiel für eine geschachtelte Nachricht. Es handelt sich um den Typ „Path“ und beschreibt einen Pfad zur Navigation bestehend aus dem Array eines weiteren Nachrichtentyps „PoseStamped“ und dem Typ „Header“.

```
#An array of poses that represents a Path for a robot to follow
Header header
geometry_msgs/PoseStamped[] poses
```

Definiert werden Nachrichtenypen über eine einfache sprach-neutrale Schnittstellenbeschreibungssprache (IDL). Mittels Code-Generatoren in der jeweiligen Programmiersprache werden daraus native Implementationen erzeugt, so dass die erzeugten Objekte, wie die nativen Objekte einer Programmiersprache nutzbar sind.

Eine besondere Funktion nimmt der Datentyp „Header“ einer Nachricht ein. Er besteht aus einer Sequenznummer, einem Zeitstempel und dem Bezeichner eines Koordinatensystems. Das Bindeglied zwischen Koordinatensystemen bildet ein ROS-Knoten für Tranformationen (*tf*).

Transformationen Bei der Entwicklung von Robotik-Anwendungen wird mit vielen Koordinatensystemen gearbeitet. Jedes Gelenk wird in ROS als der Ursprung eines eigenen Koordinatensystems beschrieben. Die Stellung eines Gelenkes und damit der Ursprung des Koordinatensystems kann sich über die Zeit verändern.
Um Transformationen zwischen Koordinatensystemen berechnen zu können ist es notwendig die Gelenkstellungen und die hierarchische Anordnung der Gelenke untereinander zu kennen. Bei komplexen Anwendungen können auch Gelenkstellungen aus der Vergangenheit von Interesse sein.
Diese Informationen stellt das ROS-Paket „tf“ dem Entwickler zur Verfügung. Es pflegt die Beziehungen der einzelnen Koordinatensysteme in einer Baumstruktur und puffert die Gelenkstellungen über ein Zeitfenster hinweg.
Erlaubt wird damit die Transformation von Punkten, Vektoren oder abstrakten Datentypen zwischen zwei beliebigen Koordinatensystemen zu einem gewünschten Zeitpunkt. Es ist dabei unerheblich, wie viele Gelenke zwischen den beiden Koordinatensystemen liegen.

Neue Informationen werden von anderen Knoten in das System eingebracht. Für gewöhnlich übernehmen diese Aufgabe Knoten, welche die Abstraktionsschicht zur Hardware bilden. Sie liefern das aktuelle Feedback in Form einer Transformation und stellen diese Information dem gesamten System zur Verfügung.

2.6.3 Themen

Mit Hilfe von Themen wird der asynchrone Versand von Nachrichten unter ROS organisiert. Knoten veröffentlichen dazu Nachrichten unter einem bestimmten *Thema*. Die Nachrichten erreichen grundsätzlich jeden Knoten, der sich für das entsprechende Thema interessiert. Dazu ist es notwendig, dass Knoten ihr Interesse am Master-Knoten anmelden.
Das Konzept erlaubt nicht nur eine one-to-many Kommunikation, sondern auch das Publizieren vieler Knoten unter dem gleichen Thema. Dies ist sinnvoll zur Fusion von gleichen Nachrichten unterschiedlicher Knoten. ROS sieht keine Mechanismen vor, mit welchen Knoten bestimmen können, wer eine Nachricht versendet oder empfangen hat.

Häufig veröffentlichen Knoten Nachrichten in einem festgelegten Intervall von zehn, zwanzig oder dreißig Herz. Es handelt sich dabei um eine Möglichkeit und keine Verpflichtung. Nachrichten können auch beim Eintreten eines Ereignisses veröffentlicht werden oder beim Vorliegen eines neuen Ergebnisses.
Wichtig ist dieser Hintergrund, um die notwendige Bandbreite einschätzen zu können, wenn man sich für ein Thema interessiert. ROS bietet ein Werkzeug namens *rostopic*, mit welchem man sich eine Übersicht über die vorhandenen Themen, Häufigkeit der Veröffentlichungen und deren Bandbreite verschaffen kann.

2.6.4 Dienste

Dienste sind eine weitere Methode des Nachrichtenaustausches und wurden für die synchrone Kommunikation eingeführt.
Das lose Publizieren von Nachrichten ist für diese Aufgabe ungünstig. Ein Dienst wird über genau ein Nachrichten-Paar definiert, bestehend aus Anfrage und Antwort. Das Verfahren entspricht einem herkömmlichen Client-Server Modell. Die Verbindung von einem Client zu einem Dienst ist persistent. Auf eine Anfrage erwartet der Client immer eine Antwort. Diese Antwort erhält deshalb ausschließlich der anfragende Client. Von Knoten angebotene Dienste werden beim Master registriert. Sie besitzen eine eindeutige Bezeichnung.

Mechanismen, die eine Dienstgüte für die Kommunikationswege gewährleisten, gibt es bisher nicht. Es wird vielmehr mit der Annahme gearbeitet, dass ausreichend Ressourcen zur Verfügung stehen. Es ist möglich Nachrichten mit einem Zeitstempel zu versehen, mit welchem sich bei Bedarf das Alter von Nachrichten feststellen lässt.

2.7 ROS auf der Android-Plattform

An der Technischen Universität München wurden Anfang 2011 erste Anstrengungen unternommen einen ROS-Knoten auf einer Android-Plattform in Betrieb zu nehmen. Für Android angepasst wurde hierbei die Python-Implementierung, welche über die Skriptsprachen-Schnittstelle SL4A (Scripting Layer For Android) [Kohb] an die Android-Plattform angebunden wurde. Bei der Schnittstelle handelt es sich um eine Abstraktionsschicht, welche es erlaubt über eine Vielzahl von Skriptsprachen auf die Programmierschnittstelle von Android zuzugreifen. SL4A stellt eine interaktive Konsole zur Verfügung, in welcher Programmcode geschrieben und getestet werden kann. Gedacht ist SL4A für Entwickler, die einfache Betriebsabläufe automatisieren möchten oder zum wiederverwerten von bestehendem Programmcode. SL4A bietet nur einen sehr eingeschränkten Zugriff auf die Android-Schnittstelle. Insbesondere fehlt die Möglichkeit komplexe Benutzerschnittstellen zu erstellen und der direkte Zugriff auf die Messdaten des berührungsempfindlichen Bildschirms.
Zur Übertragung von Sensordaten eines Android-Gerätes zu entfernten ROS-Knoten ist SL4A eine schnelle und bequeme Möglichkeit. Ebenso für die Visualisierung von

einzelnen Messwerten oder dem Senden von beliebigen Ereignissen. Für umfassendere Programmabläufe wirkt sich das Scripting Layer for Android sehr limitierend aus.

Einen weiteren Ansatz bildet das von Graylin Jay entwickelte „rosjs“ [OJC+11]. Es ist Bestandteil des rosbridge-Stacks und erlaubt den Zugriff auf eine ROS-Instanz über das Javascript-Feature websocket.
Ein ROS-Knoten stellt hierbei einen TCP-Socket zur Verfügung, mit welchem herkömmliche Webbrowser über Javascript interagieren können. Dieser Knoten bildet ausschließlich die Brücke zwischen Browser und ROS-Instanz. Er gibt gewünschte Nachrichten an den Browser weiter oder publiziert Nachrichten, die vom Browser entgegen genommen wurden. Für die Kommunikation zwischen Browser und Brücke wurden Nachrichten in der JavaScript Objekt Notation (JSON) gewählt.
Der Browser Safari, Bestandteil des iPhone-Betriebssystems ab Version 4 unterstützt die notwendigen Features zur Kommunikation mit rosjs. Für die Android-Plattform ist eine Implementierung zwar angedacht, jedoch bisher nicht erfolgt.

Auch hier wirken sich die einzelnen Komponenten auf lange Sicht nachteilig aus. Der Hauptaugenmerk bei dieser Implementierung liegt auf der Möglichkeit der Entwicklung von schnellen Prototypen mittels Javascript. Theoretisch ist es möglich den Browser samt Javascript gegen eine native Android-Anwendung zu ersetzen, welche über JSON-Nachrichten mit der rosbridge kommuniziert. Dieser Ansatz soll an dieser Stelle jedoch nicht weiter verfolgt werden, da die direkte ROS-Kommunikation ohne den Einsatz eines Vermittlers, wie die rosbridge, bevorzugt wird.

Eingesetzte Software Die Basis dieser Arbeit bildet eine native Java-Implementation eines ROS-Knotens von Damon Kohler namens rosjava [Koha]. Sie wurde am 11. Mai 2011 auf der Google I/O vorgestellt und befindet sich noch in einem Frühstadium der Entwicklung. Sie ermöglicht die direkte Kommunikation mit ROS-Knoten aus einer Android-Anwendung heraus. Durch den unmittelbaren Zugriff auf den Anwendungsrahmen und der Anwendungsentwicklung in Java sind alle Merkmale der Android-Plattform in vollem Umfang nutzbar.
Auch eine Distribution und reibungslose Installation der Anwendung über Google Play ist gewährleistet.
Rosjava erlaubt den Austausch generischer ROS-Nachrichten und damit eine nahezu lückenlose Nutzbarkeit der gesamten ROS-Infrastruktur. Es wird genutzt, um das Kamerabild des Roboters, Sensordaten und Feedback auf das Android-Gerät zu übertragen und den Roboter mit Anweisungen, wie Motorbefehlen, Zielposen zur Navigation oder einer Greifpose zu versorgen.

2.7.1 Rosjava

Rosjava ist eine ROS-Implementation entwickelt in reinem Java. Sie ist nicht zu verwechseln mit der bisher verfügbaren JNI-Version, bei der mittels Java Native Interface (JNI) eine Hülle (Wrapper) um die C++-Implementation von ROS implementiert wurde.
Neben der vollständigen Funktionalität eines ROS-Knotens werden auch die Merkmale eines Masters unterstützt. Der beschriebene Mechanismus zum Auffinden von Knoten und der Gewährleistung der peer-to-peer Kommunikation. Die Implementation nähert sich dem vollen Funktionsumfang, wie er bereits für ROS-Knoten entwickelt in C++ oder der Skriptsprache Python zur Verfügung steht. Jedoch erfährt das Projekt noch regelmäßig starke Veränderungen, so dass tiefgreifende Schnittstellen-Änderungen und veraltete oder nicht existente Dokumentation keine Seltenheit darstellen.

2.7.2 Erstellen eines Pakets

Ein ROS-Paket besteht mindestens aus einem `Manifest` und einem `Makefile`, welches den Namen des Pakets und seine Abhängigkeiten definiert. Es muss nicht unweigerlich die Quellen eines Knotens oder einer Nachrichten-Definition enthalten. ROS liefert das Hilfsprogramm `roscreate-pkg` für die einfache Erstellung neuer Pakete. Rosjava ist bereits tief in die Infrastruktur integriert. Bei der Erstellung eines neuen Pakets genügt die Angabe von Rosjava als Abhängigkeit, um die Voraussetzungen für einen in Java entwickelten ROS-Knoten zu erfüllen. Lediglich das Makefile muss um den Eintrag „`rosjava.mk`“ ergänzt werden.

Der erstmalige Aufruf von `rosmake` startet die Bootstrap-Routinen von Rosjava und sorgt für das Auflösen aller im Manifest definierten Abhängigkeiten, die für den anschließenden Kompiliervorgang notwendig sind. Neben dem Bauvorgang abhängiger Knoten werden an dieser Stelle auch Nachrichten-Definitionen generiert, so dass sie als native Java-Objekte für die Entwicklung zur Verfügung stehen.

Das Build-System Ant wird für den Erstellungsprozess von rosjava basierten Projekten eingesetzt. Nach einem einmaligen Ausführen von `rosmake` sind die notwendigen Voraussetzungen geschaffen, um ein rosjava basiertes Paket mit Hilfe von ant zu bauen. Der Erstellungsprozess enthält einen Fehler, der durch eine kritische Wettlaufsituation mehrerer Prozesse entsteht. Deshalb ist es notwendig rosmake mit der Option „`--threads=1`“ zu starten. Dies verhindert einen frühzeitigen Abbruch des Vorgangs.

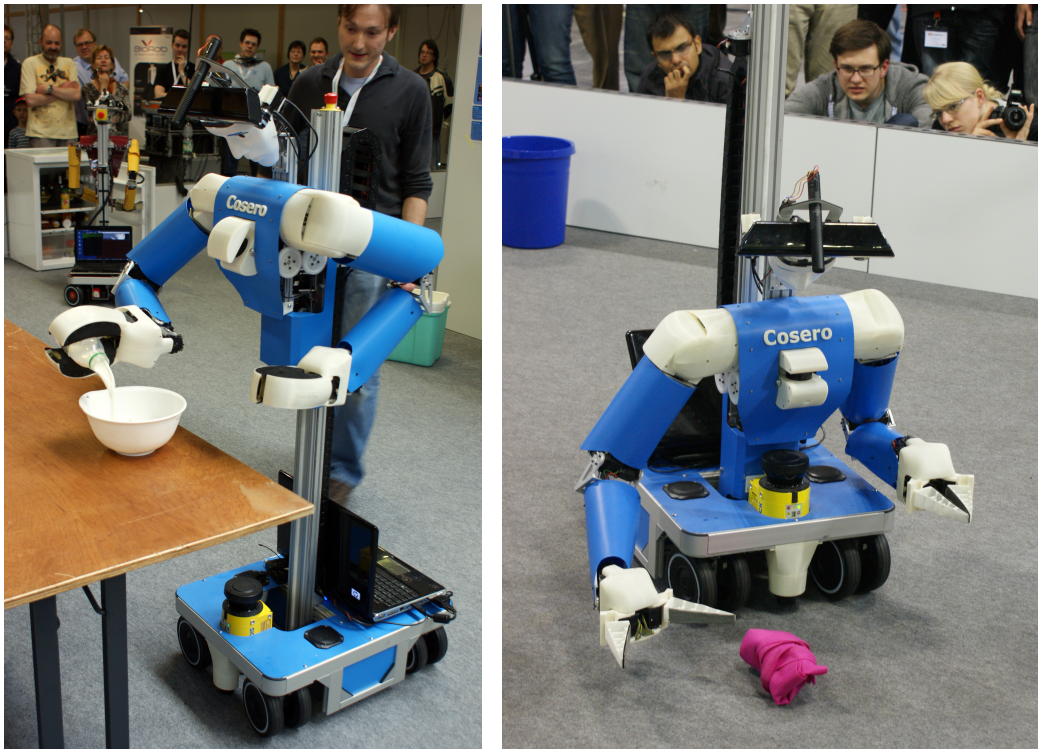
2.7.3 Erstellen von Nachrichten

Das Erstellen von Nachrichten zum Einsatz in einem Java-Projekt erfolgt durch die rosmake-Integration automatisch. Vorausgesetzt die notwendigen Pakete, welche die Nachrichten zur Verfügung stellen, wurden als Abhängigkeit im Manifest

definiert. Nach dem ersten Aufruf von `rosmake` befinden sich die Nachrichten in Form eines Archivs aus nativen Java-Objekten im Verzeichnis `.ros/rosjava/lib/`. Im Gegensatz zu den C++-Äquivalenten, in Form von Header-Dateien, werden sie nicht bei jedem Kompiliervorgang neu erstellt. Wird ein Namensraum um neue Nachrichten erweitert, so ist es notwendig, den Verzeichnisinhalt zu löschen, um aktualisierte Nachrichtenbibliotheken zu erhalten. In der Zukunft sollen die Nachrichten in einem Maven-Repository organisiert, so dass es möglich wird die Aktualität der Nachrichten zu garantieren.

Die Definition neuer ROS-Nachrichten innerhalb eines rosjava-basierenden Pakets ist bisher nicht möglich. In anderen Programmiersprachen kommt das Buildsystem *CMake* zum Einsatz, in welches das ROS-Regelwerk „`rosbuild macros`“ eingebunden werden kann. Durch den Einsatz von Ant ist dies nicht möglich.

2.8 Service-Roboter Cosero



(a) Frühstückszubereitung, Finale 2011 (b) Aufräumen, Demo Challenge 2012

Abbildung 3: Roboter Cosero bei der German Open [Beh]

Im Rahmen dieser Diplomarbeit wird mit dem kognitiven Service-Roboter Cosero der Arbeitsgruppe „Autonome Intelligente Systeme“ der Universität Bonn gearbeitet. Der Roboter dient zur Forschung im Bereich der persönlichen

Service-Robotik und wurde im Herbst 2010 in Betrieb genommen. Sowohl seine Proportionen, als auch das Erscheinungsbild sind einem Menschen nachempfunden (siehe Abbildung 3.
In Befragungen ([HHE04]) erscheint den Probanden die Größe des Roboters grundsätzlich unrelevant, sofern der Roboter klein genug ist, um in der Wohnung ungehindert navigieren zu können. Um sich an die Handlungsfreiheit eines Menschen anzunähern und eine ausreichende Integration in ein für Menschen bestimmtes Umfeld zu gewährleisten, haben sich menschliche Proportionen als vorteilhaft erwiesen. Auch ist das menschliche Erscheinungsbild ausschlaggebend, da es eine leichtere Einschätzung der Fähigkeiten des Roboters erlaubt.

Die Fortbewegung erfolgt über einen omnidirektionalen Antrieb aus insgesamt acht Rädern, die bis zu zwölf Freiheitsgrade zur Verfügung stellen. Jedes Rad wird somit einzeln angetrieben und kann in eine beliebige Richtung gedreht werden.
Es ist der einfachste Weg den Roboter wendig, zielgenau und sicher zu bewegen.

Zur Manipulation der Umwelt werden zwei leichtgewichtige anthropomorphe Arme eingesetzt. Mittels inverser Kinematik können damit Objekte gegriffen, Türen geöffnet oder Gegenstände transportiert werden. Zwei hochflexible Fingerpaare sorgen für einen stabilen und festen Griff.
Ein Aufzug erlaubt die vertikale Bewegung des Torsos. Auf diese Weise können die Einschränkungen durch das starre Rückgrat und die fehlenden Beine minimiert werden. Die Reichweite der Arme wird auf diese Weise maximiert und ermöglicht das Aufheben von Gegenständen oder das Greifen in größerer Höhe.

Die Wahrnehmung der Umgebung wird mit Hilfe von vier Laserscannern und einer RGB-D-Kamera gewährleistet. Zwei dieser Laserscanner sind im Rumpf angeordnet und dienen zur Erkennung von Hindernissen bei der Navigation, dem Verfolgen von Bein-Paaren oder ganz allgemein zur Lokalisation des Roboters in der Umgebung.
Die beiden weiteren Laserscanner sind schwenkbar am Torso positioniert. Mit ihrer Hilfe können auf einem Tisch stehende Objekte lokalisiert werden. In vertikaler Orientierung wird der Laser zur Detektion von Ebenen eingesetzt, um zum Beispiel die Höhe einer Tischplatte oder eines Regelbretts zu bestimmen. Ultraschallsensoren in den Händen unterstützen beim Greifen von Objekten.

Die RGB-D-Kamera, eine aus der Unterhaltungselektronik bekannte Microsoft Kinect, liefert neben einem gewöhnlichen Farbbild, ein mit Farbinformationen registriertes Tiefenbild. Die Auflösungen und Wiederholraten der beiden Bilder ist unabhängig wählbar. Ein in der Kamera integrierter Mikrocontroller sorgt für die Registrierung und Kalibrierung. Anwendung findet das Farb- und Tiefenbild hauptsächlich zur Objekt- und Gesichtserkennung, sowie zur Segmentierung von Ebenen oder Objekten. Die Segmentierung ist hilfreich zur Bestimmung von Objektausmaßen, Formprimitiven oder idealer Greifposen.

Die Kamera ist am Kopf des Roboters zusammen mit einem Richtmikrofon angebracht. Der Kopf ist mit Hilfe zwei weiterer Aktuatoren dreh- und neigbar.

Die Verarbeitung der Daten und Steuerung der Aktuatoren erfolgt mit Hilfe eines handelsüblichen Notebooks. Ein Intel Core i7 Prozessor stellt die benötigte Rechenleistung zur Verfügung. Die Kommunikation mit den Hardwarekomponenten erfolgt über *USB*. Die Interaktion mit dem Roboter zum Beispiel bei einer Demonstration erfolgt mittels natürlicher Sprache. Der Roboter beherrscht je nach Aufgabe ein definiertes Vokabular und eine Menge an bekannten Satzkonstruktionen.

Der Stand der Entwicklung wird jährlich in der RoboCup@Home-Liga demonstriert. Es ist der größte internationale Wettbewerb für autonome Service-Roboter und Bestandteil des RoboCups. Durch Testaufgaben und Demonstrationen wird die Performanz der einzelnen Roboter und ihrer Fähigkeiten in einer realistischen und nicht standardisierten Umgebung verglichen.
Der Wettbewerb besteht aus zwei Runden und einem anschließenden Finale. Die Bandbreite der Aufgaben erstreckt sich über das gesamte Forschungsgebiet der Robotik. Angefangen bei der Mensch-Maschine-Interaktion bis hin zur Objekt-Erkennung, Navigation und Kartierung in dynamischen Umgebungen oder der Manipulation von Objekten.

Ein Beispiel für eine übliche Testaufgabe ist „Go Get It!“. Der Roboter wird dabei aufgefordert einen Gegenstand aus einem bestimmten Raum zu holen und an den Benutzer zu übergeben. Bewertet wird dabei das erfolgreiche Auffinden des Raumes, das Auffinden und Anheben des gewünschten Gegenstandes, die erfolgreiche Übergabe an den Anwender und das Verlassen der Arena.
Eine ganz ähnliche Aufgabe ist der „Shopping Mall“-Test. Dabei wird der Roboter zu einem Regal geführt und über die enthaltenen Gegenstände aufgeklärt. Im weiteren Verlauf fordert der Anwender drei dieser Objekte an, welche der Roboter zur Kasse bringen soll.

2.8.1 Hardwareabstraktion

Nachdem ROS als Vermittlungsschicht und die eingesetzten Hardwarekomponenten des Roboters beschrieben wurden, sollen im folgenden zentrale Knoten des Service-Roboters Cosero beschrieben werden, welche zur Abstraktion der Hardware eingesetzt werden und weitere elementare Funktionen für den Betrieb des Roboters zur Verfügung stellen. Entwicklungen der Arbeitsgruppe „Autonome Intelligente Systeme“ werden in einem eigenen ROS-Stack gepflegt. Er trägt den Namen „*nimbroathome-ros-pkg*“.

Zu den Hauptkomponenten gehört der Knoten `robot_control`. Er liefert die Hardwareabstraktion zu allen eingesetzten Servos darunter auch der Servo-Anordnung, die den omnidirektionalen Antrieb bildet. Die Rückmeldung, welche die

Servos liefern, werden in Form von Gelenkstellungen auf dem Thema „joint_states“ veröffentlicht. Mit Hilfe von Komponenten des ROS-Pakets `robot_model` wird ein vollständiges Modell des Roboters in Software gepflegt. Die Beschreibung des Roboters liegt in einem URDF-Modell (Unified Robot Description Format) vor und wird zur Laufzeit geladen. Der Knoten `robot_state_publisher` übernimmt die Aufgabe die Gelenkstellungen des Themas „joint_states“ in dieses Modell einzupflegen und in einem regelmäßigen Intervall als eine Serie an Transformationen zu verbreiten. ROS ist in der Lage durch eine Visualisierung den Zustand des gesamten Roboter-Modells grafisch darzustellen. Der Hauptknoten liefert nicht nur Messwerte der Servos, sondern nimmt auch Befehle entgegen. Für die Aussteuerung der Befehle sind zwei Wege vorgesehen: Die echte Interaktion mit der Hardware oder die Übergabe der Informationen an eine Simulation. Über das Thema „drive_velocity_command“ ist die Basis über den Nachrichtentyp *Twist* einer Richtungs- und Drehgeschwindigkeit im freien Raum steuerbar.

Die Orientierung und Steuerung der einzelnen Räder im omnidirektionalen Antrieb wird vollständig maskiert. Ein Feedback des Antriebs liefert der Knoten in Form von Odometriedaten. Weitere spezielle Funktionen werden über den Dienst `bodyparts_command` zur Verfügung gestellt. Dazu gehört die Bewegung des Kopfes, die Steuerung des Aufzugs, die Auslenkung des Torsos und die Rotation des am Torso angebrachten Lasers. Die Dienste `left_arm_motion_primitive` und `right_arm_ motion_primitive` stellen eine Vielzahl von Bewegungsprimitiven der Arme zur Verfügung. Die im Verlauf dieser Arbeit genutzten Primitive lauten *convenient_pose*, *move_to_pose*, *grasp_object* und *release_object*.

Zusätzlich zu den Laserscannern wird eine RGB-D-Kamera zur Wahrnehmung der Umgebung eingesetzt. Der Standard-Treiber namens `openni_camera` liefert sowohl ein Farbbild als auch eine geordnete Punktwolke mit Tiefen- und Farbinformation.

3 Verwandte Arbeiten

In dieser Arbeit beschäftige ich mich mit der Teleoperation eines Haushaltroboters mit Hilfe eines Android-Handhelds auf der Basis von verschiedenen Autonomiestufen. Im folgenden Kapitel werden einige Projekte vorgestellt, die sich mit den einzelnen Themengebieten auseinandersetzen.

So möchte ich die Funktionalität eines autonom agierenden Butlers mit einer sehr spezifischen Aufgabe vorstellen. Anschließend werden auf Teleoperations-Anwendungen eingegangen, die bereits für das Android-Betriebssystem zur Verfügung stehen. Die beiden letzten vorgestellten Arbeiten beschäftigen sich mit sogenannten Spielzeugrobotern, die schon heute in Haushalten eingesetzt werden. Besonders hervorheben möchte ich hierbei die Möglichkeit, die Roboter mit einem bereits in den Haushalt integrierten Handheld zu steuern.

3.1 Autonomer Robotik-Butler

Bohren et al. [BRJ+11] berichten über ihre Erfahrungen und Ergebnisse beim Design und der Implementierung eines vergleichbaren Anwendungsszenarios für die Roboter-Plattform PR2 (siehe Abbildung 4). Das Hauptaugenmerk liegt dabei nicht bei der Teleoperation und der Bedienung mit Hilfe eines mobilen Gerätes, sondern auf dem Entwurf einer komplexen und robusten Roboter-Anwendung. Sie beschreiben den Aufbau einer vorbildlichen modularen Infrastruktur und schaffen die Voraussetzung für Transparenz und einfache Interaktion.

Die von ihnen betrachtete Hauptfunktionalität ist das Servieren von Getränken. Zurückgreifen können sie dabei auf bisherige Arbeiten, wie das Öffnen von Türen und Kühlschränken, der Identifikation von Gesichtern und Flaschenetiketten und der Pflege von dynamischen Hinderniskarten im dreidimensionalen Raum.

Aus der Vielzahl der Aufgaben und zu lösenden Probleme kristallisieren sie heraus, dass ein Fundus für Anwendungen geschaffen werden muss, die eine definierte Schnittstelle besitzen und unter klar definierten Randbedingungen diese erfüllen. Jede dieser Anwendungen muss ausreichend getestet sein. Eine leichte Erweiterbarkeit soll ebenfalls gewährleistet werden. Die Kapselung von Funktionalität ist bereits gängige Praxis, jedoch fehlt in der Regel eine Schnittstelle. Die Kombination dieser Komponenten soll ein robustes hetereogenes System bilden.

Sie unterteilen die eingesetzten Komponenten in drei unterschiedliche Klassen:

universelle Bestandteile bestehen aus Komponenten klassischer Robotik-Themen.

z.B. das Planen von Wegen, die Vermeidung von Hindernissen, das Berechnen der Inversen-Kinematik

Abbildung 4: Roboterplattform PR2 [Gara]

anwendungs-spezifische Bestandteile bestehen aus einzelnen Aktionen. Zur Verknüpfung dieser Aktionen dient eine State-Maschine, welche aus der nächsten Ebene gesteuert werden kann.

z.B. das Öffnen einer Türe, das Holen eines Getränks oder auch die Detektion einer Flasche.

Benutzerschnittstelle Diese Ebene wird nicht im Detail beschrieben. Aus weiteren Veröffentlichungen geht hervor, dass die Robotik-Anwendung über eine Web-Oberfläche bedient wird. Die Anwendung nennt sich BeerMe und erlaubt die Bestellung verschiedener Biere und die Lieferung an einen definierten Ort.

BeerMe Die vorgestellte Anwendung ist in der Lage einen Kühlschrank zu öffnen, die verschiedenen Biersorten zu identifizieren und gezielt aus dem Kühlschrank zu nehmen. An der mobilen Basis des Roboters ist eine Haltevorrichtung für drei Flaschen angebracht. Diese ermöglicht die simulatene Beförderung von mehreren

Flaschen ohne die Hände des Roboters dauerhaft zu belasten. Auch gibt dies dem Roboter die Möglichkeit die Arme an den Körper anzulegen oder zum Öffnen von Türen zu benutzen. Am Ziel angekommen übergibt der Roboter die Lieferung Flasche für Flasche. Für die erfolgreiche Übergabe muss die Person, welche das Bier entgegen nimmt, den Roboter ansehen. Erkennt der Roboter ein Gesicht, so lässt er die Flasche los.

Zustandsautomat SMACH Der vorgestellte hierarchische und konkurrierende Zustandsautomat namens SMACH ist die interessanteste Komponente dieses Systemvorschlags. Mit ihm werden eindeutig beschriebene Aufgabensequenzen ausgeführt. Eindeutig bedeutet, dass neben der normalen Operation auch Fehlerfälle und mögliche Lösungen explizit beschrieben sind. Ein Zustand kann mehrere Ausgänge haben, mehrer Zustände können in einem Container untergebracht werden. Auch die konkurrierende Ausführung mehrerer Zustände ist möglich. Durch diese klare Struktur war es Bohren et al. [BRJ+11] möglich Fehlerfälle einfach und effizient zu behandeln.
Evaluiert wurde mittels einer einfachen Erfolgsmetrik. Es wurden dreißig Versuche unternommen, bei welchen jeweils bis zu drei Flaschen ausgeliefert wurden.

3.2 Android-Anwendungen zur Teleoperation

Für die bereits erwähnte Roboter-Plattform PR2 findet sich in Google Play (Android Market) eine Hand voll Teleoperations-Anwendungen. Die große Anzahl sollte einen nicht verwundern, da es lediglich das modulare Konzept der eigentlichen Anwendung wiederspiegelt.
Die Hauptanwendung trägt den Namen „appchooser“. Es handelt sich um einen Client für die auf dem PR2 eingesetzte und in Abbildung 5 dargestellte Anwendungsplattform.

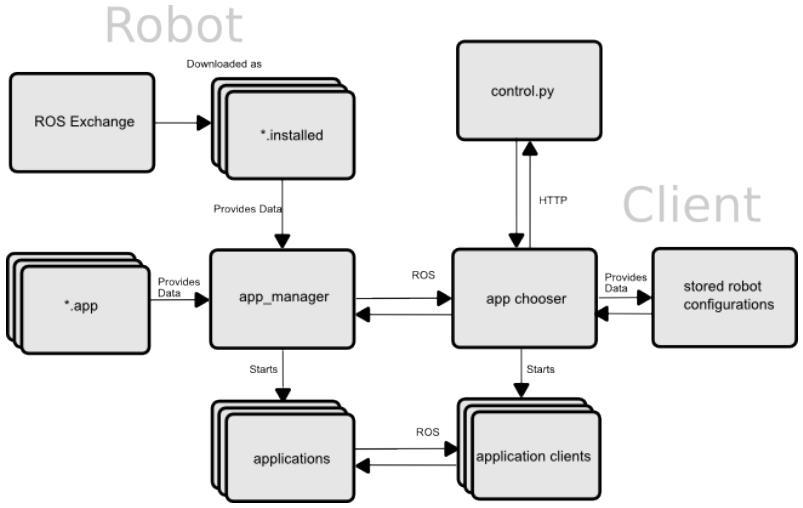


Abbildung 5: Aufbau der Anwendungsplattform [Wik]

Die Plattform erlaubt das Starten und Stoppen ganzheitlicher Robotik-Anwendungen. Eine ROS-Instanz samt Knoten wird im Normalfall über ein sogenanntes Launchfile gestartet. Unterscheiden sich zwei Robotik-Anwendungen in ihrer Natur grundlegend, so findet die Pflege in unterschiedlichen Dateien statt. Dies bietet den Vorteil, dass der Roboter nicht unnötig belastet und keine Rechenleistung verschwendet wird. Zum Wechsel einer Anwendung wird normalerweise die laufende Instanz beendet und in einer neuen Zusammenstellung nu gestartet.
Um diesen Umstand zu verbessern und die laufenden Knoten besser zu organisieren wurde die Plattform geschaffen. Sie besteht aus einem Manager-Knoten (*app_manager*), die das Starten und Stoppen ausgewählter Knoten erlaubt. Auf dem Roboter werden dazu Meta-Informationen hinterlegt, welche die Anwendungen samt zugehöriger Launchfiles beschreibt. Erhält der Verwaltungsknoten den Auftrag eine Anwendung zu beenden, so stoppt der Manager alle betroffenen Knoten und setzt den Zustand der Anwendung auf beendet.

Der *appchooser* ist somit in der Lage Funktionalität zu starten und zu beenden. Möchte ein Anwender mit dem Roboter interagieren, so startet er die Anwendung. Auf dem Roboter verfügbare Anwendungen werden in einer Auswahl angeboten. Darüber hinaus ist erkennbar, ob die Anwendung gerade läuft.

Bei der Auswahl wird auf dem Android-Gerät das passende Gegenstück gestartet. Existiert dies nicht, wird es aus dem Market nachgeladen. Dies ist der Grund für die Vielzahl der Anwendungen im Market. Verfolgt wird dabei das Android-Konzept von Intents.

3.3 WowWee Rovio

Der Rovio [Wow] der Firma WowWee ist ein preisgünstiger Spionageroboter mit omnidirektionalen Antrieb. Ausgestattet mit einer Kamera, einem Mikrofon und einer Anbindung über WLAN an das Internet ist es möglich den Roboter aus weiter Entfernung zu teleoperieren (siehe Abbildung 6).

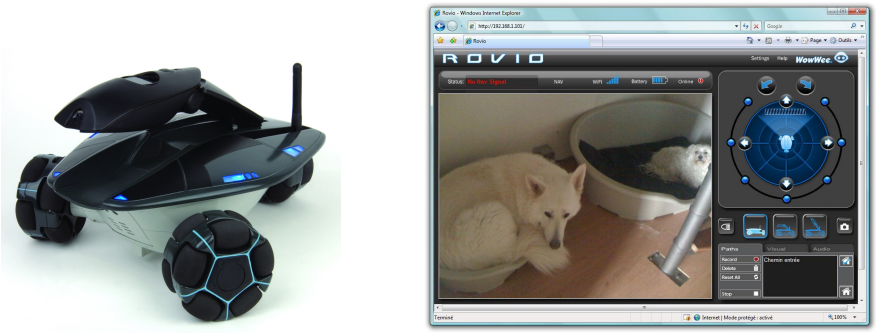


Abbildung 6: WowWee Rovio [Wow]

Geworben wird mit der ständigen virtuellen Präsenz zu Hause, der Möglichkeit sich jederzeit mit dem Roboter zu Verbinden und ihn über nur einen Klick an eine

neue Position zu schicken. Für die Lokalisierung im Raum und zum Auffinden der Ladestation wird ein System namens NorthStar eingesetzt. Es besteht aus stationären Projektoren, welche zwei Infrarotlichtkegel mit einer eindeutigen Signatur an die Raumdecke projiziert. Mittels Triangulation kann der Roboter auf diese Weise in kürzester Zeit ohne vorheriges Training oder Kartierung seine Position und Ausrichtung bestimmen. Zur einfachen Hindernisvermeidung ist der Roboter mit einem weiteren Infrarotsensor ausgestattet. Über einen Webbrowser ist es dem Anwender möglich Positionen zu speichern und Pfade zu erstellen, auf welchen der Roboter patroullieren soll. Die Kamera des Roboters befindet sich an einem Arm, welcher sich in drei Positionen steuern lässt.

Neben der Web-Anwendung zur Steuerung des Rovios existiert eine Android-Anwendung (siehe Abbildung 7), welche sich kaum im Bedienkonzept der Browser-Version unterscheidet. Auf einer Windrose lässt sich die Position des Roboters verändern, indem man ihn in eine gewünschte Richtung aus dem Zentrum zieht. Drehungen um die eigene Achse werden über zwei Buttons ausgeführt, ebenso wie die Steuerung der Kameraposition und die Aktivierung einer Lichtquelle. Es ist möglich Bildschirmfotos aufzunehmen, gespeicherte Pfade abzulaufen oder über einen Panikknopf den Roboter zum Halten zu zwingen.

Abbildung 7: Androvios Android-Anwendung [Prod]

3.4 Parrot AR.Drone

Die Parrot AR.Drone [Par] ist ein Quadrocopter konzipiert zur Steuerung über ein iPhone oder einem anderen iOS-Gerät (siehe Abbildung 8).

Mittels erweiterter Realität wird eine Spielplattform geschaffen, in welcher Dronen gegeneinander antreten können. Auf der Internationalen Funkausstellung 2011

Abbildung 8: Parrot AR.Drone [PR]

wurde außerdem eine Android-Version der AR.FreeFlight-Anwendung vorgestellt, sowie ein Software Development Kit zur Weiterentwicklung dieser. Auf die Fahnen geschrieben hat sich Parrot eine intuitive und benutzerfreundliche Steuerung.

Die Drone ist mit zwei Kameras ausgerüstet. Die vertikale Hochgeschwindigkeitskamera ist gemeinsam mit einem System zur Trägheitsmessung verantwortlich für Geschwindigkeitsmessung und die Stabilisierung in der Luft. Die „Parrot Smart Piloting“-Technologie kompensiert Turbulenzen bei Außenflügen. Der Ultraschallhöhenmesser sorgt für einen gleichmäßgen Abstand zu Boden oder Hindernissen. Die Front-Kamera überträgt das Livebild zum Anwender. Eingesetzt wird ein WLAN basiertes Ad-hoc-Netzwerk, welches sich in der Realität als nachteilig herausgestellt hat. Ein Infrastruktur-Modus wurde nachgerüstet, über welchen die Drone an WLAN-Zugangspunkte angemeldet werden kann. Ein Roaming zwischen Zugangspunkten ist somit möglich.

Die Angebotsvielfalt an Steuerungssoftware für Geräte mit berührungsempfindlichen Bildschirmen nimmt laufend zu. Verfolgt werden dabei zwei unterschiedliche Ansätze zur Steuerung. Entweder erfolgt die Steuerung über zwei virtuelle Joysticks, welche transparent auf dem Livebild angeordnet sind oder einem einzelnen für die Steuerung von Rotation und Flughöhe und den Lagesensor des Geräts für die Neigung der Drone. Letzterer wird erst aktiviert, wenn man einen Knopf dauerhaft mit einem Finger berührt. Sobald man den Finger vom Display des Gerätes entfernt übernimmt ein Autopilot und sorgt für Stabilität und versetzt die Drone in einen Schwebeflug. Gleiches Verhalten setzt ein, sollte die WLAN-Verbindung abreißen oder der Anwender einen Anruf entgegen nehmen. Die Steuerungselemente reagieren optisch auf Eingaben und vermitteln das Gefühl von Ansprechbarkeit.

Abbildung 9: Parrot FreeFlight

4 Entwurf der Anwendung

Durch die Veränderung der Beziehung zwischen Mensch und Roboter und der wachsenden Handlungsvielfalt sind neue Anforderungen für eine Robotik-Anwendung entstanden. Das angestrebte Interaktionsmodell hat sich von einer Master-Slave-Beziehung zu der eines gemischten Teams entwickelt. Anwender und Roboter kommunizieren miteinander, um ein gemeinsames Problem zu lösen und gegenseitig zu assistieren. Im Idealfall lernt jedes der Teammitglied im Laufe der Zeit dazu, so dass die Bindung und das Vertrauen wächst.

Ein Roboter soll nicht länger nur von einem Experten bedienbar sein. Beliebigen Personen unabhängig von ihren Vorkenntnissen soll es ermöglicht werden einen Roboter zu bedienen. Die Belastung des Anwenders und die benötigte Vorbereitungs- und Ausführungszeit für eine Aufgabe soll so gering wie möglich sein.
Die in dieser Arbeit mit Hilfe von Autonomie und einer geschickten Benutzeroberfläche zu erzielende Erleichterung soll sich nicht einschränkend auf die Handlungsfreiheit des Roboters auswirken.

4.1 Anforderungen

Im Folgenden werden die Anforderungen an die Anwendung explorativ bestimmt. Dabei werden berücksichtigte Prinzipien aus verwandten Arbeiten und Themenkomplexen erläutert und adaptiert. Um eine Vollständigkeit zu gewährleisten wird ein Vergleich mit den Anforderungen an die Teleoperation in der Rettungsrobotik unternommen. Sowohl gleiche als auch konträre Anforderungen werden dabei aufgezeigt und eingeordnet.

Bei der in dieser Arbeit entwickelten Benutzeroberfläche wird besonderes Augenmerk darauf gelegt, dass der Anwender in seiner Handlungsfreiheit nicht eingeschränkt wird.
Stellt man sich vor, dass ein Roboter vollständig autonom agiert und jede Möglichkeit in eine Handlung einzugreifen fehlt, so wird die Selbstständigkeit des Anwenders nur sekundär gesteigert. Der Anwender ist weiterhin künstlich in seinen Handlungen eingeschränkt. Auch der gegensätzliche Ansatz dem Roboter keinerlei Autonomie zu verleihen, ist kein idealer Ansatz. Eine direkte Steuerung des Roboters kostet große Aufmerksamkeit, ist fehleranfällig und zeitintensiv.
Eine harte Anforderung eines persönlichen Service-Roboters ist deshalb die einfache Bedienbarkeit durch einen Laien. Der Anwender darf nicht gezwungen werden sich das Wissen eines Experten aneignen zu müssen.

Um die Fähigkeiten und Stärken von Anwender und Roboter auszunutzen, wird der Ansatz eines gemischten Teams verfolgt. Nicht nur der Roboter steht dem Anwender unterstützend zur Seite, auch der Anwender unterstützt den Roboter. Über diesen Weg können künstliche Grenzen überschritten werden.

Aufgaben der Bilderkennung und des Bildverstehens sind für den Menschen keine große Herausforderung, für einen Roboter hingegen schon. Erreicht der Dienstleistungsroboter die Grenzen seiner kognitiven Fähigkeiten, kann der Anwender dem Roboter assistieren und bei der Lösung für ihn schwieriger Aufgaben behilflich sein.
Die Stärke des Roboters ist die parallele Verarbeitung großer Mengen an diskreten Sensordaten, die ihn in die Lage versetzten potentielle Gefahrensituationen zu erkennen, bevor der Anwender sich diesen bewusst ist. Er kann Hindernisse melden oder autonom umfahren, auf die ein möglicherweise nachlässiger Anwender zusteuert.

Die Anwendung soll deshalb die Steuerung des Roboters auf drei unterschiedlichen Autonomiestufen erlauben. Auf diese Weise ist es möglich sowohl zuvor nicht spezifizierte Vorgänge manuell auszuführen, als auch vollständig definierte Aufgaben umzusetzen. Die unterschiedlichen Abstraktionsschichten tragen Sorge, dass die Belastung des Anwenders minimal gehalten wird, sofern es die Situation erlaubt.

Olsen und Goodrich [GO03] haben einen Leitfaden aus sieben Prinzipien zur effiziente Mensch-Roboter-Interaktion entworfen, welcher auf Basis eigener Erfahrungen, Experimenten und Eindrücken entstanden ist. Die für diese Anwendung wichtigsten Prinzipien werden im Folgenden beschrieben und berücksichtigt.

Das Prinzip „Manipuliere die Welt anstelle des Roboters“ hat den Entwurf der Anwendung besonders beeinflusst. Es ist notwendig sich vor Augen zu führen, dass nicht der Roboter im Mittelpunkt der Handlung steht, sondern die Handlung an sich. Das Ziel ist die erfolgreiche Umsetzung einer Aufgabe und damit die Manipulation der Welt.
Selbst in der geringsten Autonomiestufe findet dieses Prinzip Anwendung. Möchte ein Anwender mit Hilfe eines Roboterarms einen Gegenstand greifen, so ist die Steuerung jedes einzelnen Gelenkes nur das Mittel zum Zweck. Das abstrakte Ziel ist jedoch das Erreichen einer gewünschten Endeffektorpose in der Welt. Nach Olsen und Goodrich soll deshalb die Manipulation dieser Pose im Vordergrund stehen.
Natürlich werden mit diesem Ansatz Einschränkungen in Kauf genommen. Diese können mit Hilfe von Autonomie und den kognitiven Fähigkeiten des Roboters aufgefangen werden. Vorteilhaft ist die Vermeidung von Selbstkollisionen oder die Berücksichtigung von Hindernissen auf dem Weg zum Ziel. Steht der Roboter dem zu greifenden Objekt ungünstig gegenüber, so sollten Nebenbedingungen in die Berechnung der Greifbewegung einfließen. Alternativ ist es möglich den Benutzer über die Anwendung unterstützende Lösungen, wie eine Neuausrichtung der Basis oder Verschiebung des störenden Gegenstandes, anzubieten. Letzteres verhindert Fehleinschätzungen des Benutzers und bewahrt vor den Auswirkungen einer fehlerhaften Wahrnehmung des Roboters.

Nicht berücksichtigt werden konnte das sechste Prinzip von Olsen und Goodrich [GO03], die Externalisierung des Gedächtnisses.
In einem Beispiel wird das eingeschränkte Blickfeld der Kamera angereichert, indem in der Vergangenheit wahrgenommene Objekte mit ihrer räumlichen Beziehung vermerkt werden. Eingesetzt werden kann dazu ein exozentrischer Blickwinkel der Kamera. Die Kamera befindet sich dabei ein Stück hinter dem Roboter und füllt nur einen Teilbereich des Bildschirms aus. Objekte, die bereits aus dem Sichtfeld der realen Kamera gerückt sind werden in der virtuellen Szene weiterhin dargestellt.
Auch die Messreihen von Laserscannern können in dieser Darstellung für den Anwender einfach verständlich integriert werden. Dies bietet einen entscheidenden Vorteil: Dem Anwender ist es möglich auf einen Blick eine umfassende Übersicht über die Szene zu erhalten. Es ist keine Interaktion notwendig, um die aktuelle Situation einzuschätzen. Vorausgesetzt, es hat bereits eine ausreichende Exploration stattgefunden.
Der Anwender steht nicht länger in der Pflicht sich die Umgebung einzuprägen. Besonders, wenn der Anwender dem Roboter nicht seine volle Aufmerksamkeit widmet oder kurzzeitig abgelenkt ist, fällt es ihm einfacher ein Situationsbewusstsein zu erlangen. Dieser Ansatz schont nicht nur das Gedächtnis, sondern löst ein weiteres entscheidendes Problem der Teleoperation: Das oft beklagte eingeschränkte Sichtfeld und der damit resultierende große tote Winkel beeinträchtigt nicht länger die Wahrnehmung des Anwenders. Dieser besitzt durch den exozentrischen Blickwinkel ein Gefühl für die Ausmaße des Roboters und erhält einen direkten Einblick in die unmittelbare Umgebung des Roboters.
Vergleicht man die Vorteile dieser Lösung mit dem menschlichen Bewusstsein, so erweist sie sich als äußerst unterstützend. Das Sichtfeld eines Menschen besitzt ebenfalls einen toten Winkel. Wahrnehmungen innerhalb dieses Bereiches werden von anderen Sinnen erfasst und soweit wie möglich aufgefüllt. Besitzt ein Mensch ein geschränktes Wahrnehmungsvermögen oder ist seine Aufmerksamkeit gemindert, so müsste er diesen toten Winkel regelmäßig kontrollieren, um Gefahren auszuschließen.

Viele Ausarbeitungen der Mensch-Roboter-Interaktion beschäftigen sich mit dem Entwurf und der Auswertung von Teleoperationsanwendungen aus der Rettungsrobotik. Im Folgenden soll aufgezeigt werden, in welchen Disziplinen sich die Anforderungen dieser Anwendungen unterscheiden oder gleichen.

Anwender Der Operator eines Rettungsroboters entspricht dem vollständigen Gegenteil von dem eines persönlichen Service-Roboters. Er ist ausgebildet und wurde für den Betrieb des Roboters geschult.
Für gewöhnlich bedienen gleich zwei Personen einen einzelnen Roboter. Die Gründe dafür sind vielfältig: Die Arbeitsbelastung für die Operatoren ist überdurchschnittlich hoch, die Umgebung ist nur schwer zu erfassen, Kartenmaterial ist unbrauchbar und die Zugänglichkeit von Räumen muss mühsam erarbeitet werden. Auch die Froschperspektive auf Grund der möglichst geringen Größe des Roboters

kann sich beeinträchtigend auswirken. Der Roboter ist der Gefahr ausgesetzt, verloren zu gehen. Außerdem kann eine Fehlbedienung die Situation verschlimmern und Menschenleben kosten.
Der erste Operator ist für die Navigation verantwortlich, um eine sichere Bewegung durch einen zum Beispiel einsturzgefährdeten Korridor sicherzustellen. Er muss sich dabei einen Eindruck über den Untergrund verschaffen und sicherstellen, dass der Roboter nicht abstürzt oder durch andere Umstände verloren geht.
Der zweite Operator besitzt die Aufgabe die Umgebung zu inspizieren und Überlebende ausfindig zu machen. Ihm stehen dabei spezielle Sensoren zur Verfügung, die Wärmesignaturen oder einen CO2-Ausstoß sichtbar machen.
Auf Grund der Ausnahmesituation und dem Risiko Menschenleben zu gefährden steht der Operator unter besonderem Stress. Jede Handlung muss sorgfältig überlegt sein. Eine falsche Bedienung kann dazu führen, dass der Roboter verschüttet wird, sich fest fährt oder auf einem anderen Weg verloren geht.
Auch die Absichten und Ziele eines Anwenders werden unterschiedlich charakterisiert. Die Ziele des Anwenders eines Rettungsroboters ist die Rettung von Leben. Der Anwender trägt eine hohe Verantwortung und steht unter einer entsprechenden Belastung. Bei der Übung von Gefahrensituationen und dem Training des Anwenders ist es schwer eine vergleichbare Situation zu schaffen.

Die Ziele in der persönlichen Service-Robotik sind im Vergleich gefahrlos. Der Roboter dient zur Unterstützung. Die Brisanz der Aufgaben ist jederzeit überschaubar und der in seiner Mobilität eingeschränkte Anwender kann sich im Notfall einen Dritten zur Hilfe holen, sollte eine Handlung missglücken oder der Roboter auf Grund ungünstiger Umstände vorübergehend verloren gehen.
Die Aufgabe des Roboters ist nicht das Bringen und Verabreichen von lebenswichtigen Medikamenten, sondern die Erledigung von alltäglichen unkritischen Aufgaben. Der Stress, den ein Anwender bei der Nutzung eines persönlichen Service-Roboters ausgesetzt ist, ist deshalb abhängig von der jeweiligen Aufgabe. Aktivitäten können jederzeit unterbrochen werden.
Ist der Anwender nicht körperlich eingeschränkt und hilft der Roboter bei alltäglichen Aufgaben, so würde man den Nutzen des Roboters als Luxus beschreiben. Aufgaben, die nicht erledigt werden, bleiben liegen und müssen gegebenenfalls anders gelöst werden.

Erlernbarkeit Für den Betrieb von Rettungsrobotern werden die Operatoren geschult, um in Gefahrensituationen eine effektive und effiziente Arbeit zu leisten. Damit keine Opfer übersehen werden und der Roboter möglichst nicht verloren geht, werden umfangreiche Informationen über die Umwelt geliefert. Eine Visualisierung von Laserscans, Messdaten von Ultraschallsensoren, Wärmebilder, CO2-Ausstoß, ein Farbbild einer oder mehreren Kameras, eine Karte von bereits explorierten Bereichen, den Ladestand des Akkus, die Qualität der drahtlosen Anbindung und vieles mehr. Jede dieser Informationen muss vom Anwender verarbeitet und interpretiert werden. Umso schlechter die Repräsentation dieser Daten, desto höher steigt die Belastung des Anwenders.

Mit Hilfe eines speziellen Trainings lernen die Operatoren diese Daten zu interpretieren, die Fähigkeiten des Roboters einzuschätzen und Gefahren zu erkennen.

Auch in der persönlichen Service-Robotik ist es nicht ausgeschlossen, dass Anwender eine Schulung erhalten. Er soll dabei jedoch nicht zu einem Spezialisten ausgebildet werden. Vielmehr ist es notwendig die Fähigkeiten des Roboters einschätzen zu lernen und ein Vertrauensverhältnis zu entwickeln.
Um den Anwender zu entlasten soll die Benutzeroberfläche nur so viele Informationen, wie gerade notwendig liefern. Die Repräsentation dieser Informationen soll dabei natürlich und leicht verständlich sein. Möglich wird die Reduzierung des Informationsangebots durch die Steigerung der Autonomie des Roboters, so dass weiterhin eine gefahrlose Interaktion mit seiner Umgebung gewährleistet werden kann.
Das Situationsbewusstsein des Anwenders nimmt damit in der Service-Robotik ebenfalls einen anderen Stellenwert ein. Es besteht keine Gefahr, dass der Roboter für immer verloren geht. Die eingeschränkte Wahrnehmung für den Anwender gefährdet keine Menschenleben. Auf eine Information über CO2-Ausstoß darf gänzlich verzichtet werden, auch Wärmebilder bieten nur in seltenen Fällen einen Mehrwert.
Informationen, die der Roboter zur Verfügung stellt und in Einzelfällen hilfreich sein können, sollten nur optional eingeblendet werden, aber kein dauerhafter Bestandteil der Benutzeroberfläche sein. Auf diese Weise ist es dem Anwender möglich sich über einen längeren Zeitraum an den Roboter und seine Bedienung zu gewöhnen. Seine Funktion kann langsam und spielerisch erkundet werden bis die Stärken und Schwächen des Roboters bekannt sind.
Ebenso wie bei erfahrenen Operatoren von Rettungsrobotern wird die Performanz bei der Arbeit mit dem Roboter mit der Zeit wachsen. Voraussetzung für einen Lernprozess ist eine gute Benutzbarkeit des Bedienoberfläche.

Umgebung Die unstrukturierte und nur schwer einschätzbare Umgebung in der Rettungsroboter eingesetzt werden ist der Hauptgrund, weshalb die Steuerung eines Rettungsroboters Experten erfordert. Mit den Mitteln, die die künstliche Intelligenz heutzutage zur Verfügung stellt, ist es nicht möglich die Befahrbarkeit eines Untergrunds universell zu beurteilen, die Stabilität von Stahlträgern einzuschätzen oder ein zerstörtes Objekt zu identifizieren und daraus Folgerungen abzuleiten. Für einzelne Fälle lassen sich heute schon Lösungen erarbeiten, jedoch ist in der Rettungsrobotik ein umfassendes Bewusstsein über die aktuelle Situation notwendig, um Fehlentscheidungen auszuschließen. Fehlerhafte Annahmen können die Situation verschlechtern oder das Vertrauen in den Roboter beeinträchtigen. Dies hat den geringen Einsatz von Autonomie und eine entsprechende Belastung des Anwenders zur Folge. Der Mangel an Autonomie muss vom Benutzer kompensiert werden.

Die Arbeitsumgebung eines persönlichen Service-Roboters ist im Vergleich zu der eines Rettungsroboters strukturiert.

Es dürfen eine Vielzahl von Annahmen über die Umgebung unternommen werden. Dazu zählt die Beschaffenheit des Untergrunds, zum Boden parallele Ablageflächen, wie Regale oder Tischplatten oder die gleichbleibenden Ausmaße von Räumen.
Der Anspruch an Mobilität und die Erreichbarkeit eines jeden Ortes ist nur eine von vielen Disziplinen, die ein Service-Roboter in seiner Umgebung leisten soll. Es müssen deshalb Einschränkungen akzeptiert werden, um eine ausgewogene Lösung zu ermöglichen. Das Umfeld, in welchem der Roboter arbeitet ändert sich für gewöhnlich nur marginal. Diese vergleichsweise geordneten Umstände erlauben und erleichtern das Erlernen einer Umgebung. Der Roboter kann die Fähigkeit erhalten seine Position in seiner Umgebung zu bestimmen. Er kann gezielt Räume aufsuchen oder dynamische Hindernisse identifizieren und gefahrlos umfahren. Auch das Erlernen von Gegenständen ist eine Option und wird auf Dauer die Arbeit mit dem Roboter erleichtern.

Situationsbewusstsein Auch das Situationsbewusstsein nimmt in der Rettungs- und Service-Robotik einen unterschiedlichen Stellenwert ein.
Der Operator eines Rettungsroboters ist für die vollständige Einschätzung der Situation verantwortlich. Ihm darf kein Detail aus der Umgebung des Roboters entgehen. Während des gesamten Betriebs besteht die Gefahr einen Menschen zu übersehen oder den Roboter zu verlieren. Das wiederholte Aufsuchen eines Ortes ist ebenfalls mit Risiko und dem Verlust von Zeit verbunden.
In der persönlichen Service-Robotik sind Zeit und Gefahr weniger kritische Faktoren. Gefahren können mit Hilfe von Autonomie reduziert werden. Einen Großteil des Situationsbewusstseins kann deshalb der Roboter leisten. Mit Hilfe eines akkuraten Modells von sich selbst, ist es ihm möglich Kollisionen zu vermeiden. Nicht nur eine robuste und gefahrlose Fortbewegung wird durch diesen Ansatz erlaubt, auch eine gefahrlose Interaktion in der Umgebung wird ermöglicht. Es muss dabei sichergestellt werden, dass der Roboter weder sich, noch der Umwelt Schaden zufügt.

Für die Aufgaben, die ein persönlicher Roboter leisten soll konzentriert sich die Notwendigkeit des Situationsbewusstseins des Anwenders zumeist auf einzelne kleine Szenen.
Menschen suchen Gegenstände an bestimmten Orten, wo sie typischerweise Dinge ablegen, sie gut aufgehoben erscheinen oder zuletzt gesehen wurden. Es genügt deshalb, wenn der Roboter den Einblick auf ein einzelnes Regalbrett oder einen Schreibtisch liefert. Orte von Interesse werden unweigerlich in den Fokus der Kamera gerückt.
Ausgenutzt wird dabei die Vertrautheit der eigenen Wohnung. Der Roboter muss nicht für einen ganzheitlichen Überblick sorgen. Tote Winkel auf Grund eines eingeschränkten Sichtfeldes der Kamera können vernachlässigt werden, solang die Autonomie des Roboters die Erkennung von Gefahren erlaubt. Auch die Zuhilfenahme von unabhängigen Kameras für einen Einblick aus einer weiteren Perspektive ist vorstellbar.

Das Bewusstsein, welches dem Anwender eines persönlichen Service-Roboters verborgen bleibt, muss der Roboter ausgleichen. Werden in Folge dessen Aktionen verweigert oder abgeändert, muss dies für den Anwender transparent erfolgen. Insbesondere bei nicht offensichtlichen Hindernissen oder Problemen ist der Anwender zu informieren. Kann der Anwender die Entscheidungen des Roboters nicht nachvollziehen, so neigt er dazu Schutzmechanismen zu deaktivieren und den Roboter erhöhter Gefahr auszusetzen. Die in Kauf genommenen Einschränkungen für eine bessere Übersicht und die leichtere Erlernbarkeit wirken sich schnell kontraproduktiv aus.

Gemeinsamkeiten Die Fähigkeiten eines Rettungsroboters an Hand seiner äußeren Erscheinung einzuschätzen und Gemeinsamkeiten festzustellen erweist sich oft als schwierig. Rettungsroboter können vereinzelt kopfüber betrieben werden. Die Größe oder die Anzahl und Position der Räder kann einen Hinweis darauf liefern, jedoch ist es für den Anwender schwierig derartige unnatürliche Fähigkeiten zu erkennen. Aber auch einfache Sachverhalte können, selbst für einen erfahrenen Anwender, schwierig festzustellen sein. Die einzelnen Sensoren und deren Aufgabe sind nur für geschulte Augen kategorisierbar. Gemeinsamkeiten bei einem persönlichen Service-Roboter mit menschlichen Proportionen und Erscheinungsbild festzustellen, ist weitaus leichter. Insbesondere, wenn der Anwender mit einfachen und natürlichen Hinweisen versorgt wird, werden die Fähigkeiten des Roboter schnell transparent und einschätzbar.

Die Interaktion mit dem Roboter soll so natürlich, wie nur möglich erfolgen.
Möchte der Anwender einen Gegenstand vom Boden aufheben, so sollte es nicht notwendig sein, den Roboter anzuweisen in die Beuge zu gehen, um den gewünschten Gegenstand in Reichweite der Arme zu befördern und ihn anschließend zu greifen.
Wird die Endeffektorpose in Bereiche außerhalb der Reichweite der Hände gesteuert, so muss der Roboter Vorschläge für das Erreichen dieser Position liefern. Dies kann das Angebot sein, in die Beuge zu gehen, indem der Aufzug nach unten gefahren wird oder die Position der Basis zu korrigieren, um einen Gegenstand erreichen zu können.
Ein passiver Ansatz, um den Anwender frühzeitig auf die Problematik aufmerksam zu machen, ist das visuelle Kenntlichmachen des Einzugsbereichs der Arme. Es handelt sich dabei um Probleme, die durch die verborgene Absicht des Anwenders entstehen und effizient gelöst werden müssen. Wäre die Absicht und damit das Ziel vor Beginn der Handlung eindeutig, so könnte die Autonomie für eine korrekte Positionierung sorgen.

Gertman und Brümmer [Ger08] empfehlen die Präsentation von abstrakten Informationen, um die Wahrnehmung zu verbessern.
Vergleicht man die Rohdaten eines Laserscans in Tabellenform mit einer Visualisierung der Daten mit Hilfe einer gezeichneten Silhouette, so kann man dieser

Empfehlung nur zustimmen. Jedoch sollten die präsentierten Sensordaten auch frei von mentalen Transformationen in die Benutzeroberfläche integriert werden. Häufig werden sie aus dem ursprünglichen Kontext herausgelöst dargestellt. Der Benutzer benötigt dabei das Wissen über die Lage, Position und Orientierung des Laserscanners, um die eigentliche Information interpretieren zu können. Umso mehr unterschiedliche Komponenten der Anwender zu kontrollieren hat, desto höher die notwendige Konzentration. Die eingesetzte Visualisierung soll deshalb natürlich und leicht verständlich sein.

Werden die gleichen Messungen perspektivisch in Form eines farbigen Bandes in ein Kamerabild projiziert, so wird es möglich diese Information nahezu unterbewusst zu verarbeiten. Auf Grund der dreidimensionalen Darstellung wird sogar die relative Lage des Scanners ersichtlich. Auch Kartenmaterial kann auf diese Weise eingebettet werden, um die Orientierung des Anwenders verbessern.

Berücksichtigt man in diesem Zusammenhang, dass der Roboter in der eigenen Wohnung eingesetzt wird, so ist der Mehrwert dieser Informationen bedenklich. Überhaupt muss sich bei der Visualisierung von Sensordaten die Frage gestellt werden, mit welchem Ziel einem Anwender die Daten zur Verfügung gestellt werden. Sowohl die Vielfalt der eingebetteten Informationen, als auch die Anzahl der Bedienelemente soll so minimal wie nur möglich gehalten werden. Dies trägt dazu bei, dass besondere Ereignisse, die nicht die Unterbrechung der aktuellen Handlung zur Folge haben, ausreichend Aufmerksamkeit auf sich ziehen und nicht unbeachtet bleiben.

4.2 Drahtlose Kommunikation

Die Android-Plattform erfordert keine Einschränkungen bei der Wahl eines Kommunikationsweges. Es liegt vielmehr in der Hand des Geräteherstellers mit welcher Funktionalität ein Endgerät ausgestattet wird. Zu den gängigsten integrierten drahtlosen Kommunikationswegen gehören Bluetooth, Mobilfunk und Wireless LAN, ein Standard für drahtlose Netzwerke.

Die Kommunikation mit einem Roboter über Bluetooth in einem so genannten „Personal Area Network“ ist grundsätzlich möglich, jedoch ist dieser Kommunikationsweg sowohl in seiner Reichweite, als auch durch einen niedrigen Datendurchsatz von maximal $732,2$ kbit/s stark limitiert. Die Reichweite ist für den Sichtkontakt konzipiert und wird überlicherweise eingesetzt, um kurzfristig kleine Datenmengen, wie Visitenkarten, Klingeltöne oder Bilder zu übertragen. Die Anforderungen für die Übertragungen eines Kamerabildes aus entfernten Räumen erfüllt Bluetooth nicht.

Die Verwendung einer Datenverbindungen über ein Mobilfunknetz ist ebenfalls nur mit Einschränkungen möglich. Das Problem stellt in erster Linie die peer-to-peer Architektur von ROS dar, dessen Nutzung nur wenige Anbieter ermöglichen. Die

Großzahl der Mobilfunkanbieter vergeben Endgeräten keine exklusive IP-Adresse, sondern nutzen ein Verfahren namens „Network Address Translation“ (NAT), bei welchem viele Endgeräte mit einer einzelnen IP-Adresse maskiert werden. ROS-Knoten setzen jedoch voraus, dass eine direkte Kommunikation zu einem vom Master-Knoten angekündigten Port möglich ist. Mit Hilfe eines virtuellen privaten Netzwerkes (VPN) lässt sich diese Problematik umgehen.
In diesem Fall wirkt sich nur noch die Latenz und die Bandbreite als limitierender Faktor aus. Durch die steigende Verbreitung von Long-Term-Evolution (LTE), einem Mobilfunkstandard und UMTS-Nachfolger, der besonders geringe Latenzen und damit die Übertragung von Sprachdiensten oder Videotelefonie massentauglich machen soll, wird dieses Limit bald der Vergangenheit angehören.

Für diese Arbeit wurde WLAN als das bevorzugte drahtlose Kommunikationsverfahren gewählt. Die Standards *802.11 b/g/n* sind weit verbreitet und genießen große Akzeptanz. Kaum ein Smartphone, geschweige denn Notebook, wird in der heutigen Zeit ohne WLAN-Unterstützung ausgeliefert.
Der hauptsächliche Unterschied der Standards liegt im möglichen Datendurchsatz. Im Idealfall stehen dem Anwender eine Bandbreite von 600 MBit/s brutto zur Verfügung. Dies setzt jedoch voraus, dass beide kommunizierenden Geräte vier Sendeempfangseinheiten besitzen. Pro Einheit wird dabei ein Durchsatz von 150 MBit/s ermöglicht. Der Standard 802.11*g* bietet eine Bandbreite von 54 MBit/s brutto, wobei man effektiv mit vierzig Prozent dieser Bandbreite rechnen darf. Betrieben werden drahtlose Netzwerke dieser Art in zwei unterschiedlichen Modi.

Ad-Hoc-Modus Im Ad-Hoc Betrieb können Endgeräte kurzfristig ohne weitere Infrastruktur verbunden werden. Dieser Modus bietet sich an, wenn spontan eine Netzwerkverbindung zwischen zwei Endgeräten aufgebaut werden soll. Trotz grundsätzlicher Unterstützung von gängigen Betriebssystemen hat dieser Modus beim Endanwender keine große Popularität erfahren. Dies ist möglicherweise ein Grund, weshalb er auf Android-Geräten lange vernachlässigt wurde und erst seit kurzem offiziell unterstützt wird.

Infrastruktur-Modus Beim Infrastruktur-Modus existiert ein zentraler Zugangspunkt, der seine Umgebung mit einem drahtlosen Netzwerk versorgt, an welches sich Endgeräte anmelden. Durch einen transparenten Wechsel zwischen Zugangspunkten ist es möglich die Reichweite eines Funknetzes auszudehnen, so dass eine flächendeckende Kommunikation in einem gemeinsamen physikalischen Netzwerk gewährleistet werden kann.
Die Verbreitung dieses Standards ist durch den Vertrieb in Verbindung mit einem Internetanschluss enorm und in kaum einem Haushalt mehr weg zu denken. Ein solches drahtloses Netzwerk bildet die Basis zur Kommunikation zwischen Handheld und persönlichem Service-Roboter. Der Roboter kann einen größeren Gebäudekomplex erkunden, ohne das dafür Sorge getragen werden muss, dass er sich dauerhaft im Einzugsbereich von ein und demselben Zugangspunkt befindet.
Bei Rettungsrobotern stellt das Abreißen der Kommunikation ein ernstzunehmen-

des Problem dar. Dies kann zu einem dauerhaften Verlust des Roboters führen. Der Roboter wird in diesen Fällen mit einer Notfallstrategie ausgestattet, die es dem Roboter erlauben soll autonom an den Ort des letzten Kontakts zurück zu kehren. Im Folgenden wird vorausgesetzt, dass der Roboter den Einzugsbereich des drahtloses Netzwerkes nicht verlassen kann. Um diese Limitierung aufzuheben ist es möglich eine Karte zu pflegen, in welcher die Signalausbreitung der Zugangspunkte vermerkt werden. Der Roboter kann so kritische Bereiche identifizieren, umgehen oder frühzeitig melden.

4.3 Autonomiestufen

Die Anwendung erlaubt dem Benutzer die Interaktion mit dem Roboter auf drei unterschiedlichen Autonomiestufen. Die Stufen wurden logisch voneinander getrennt, um den Anwender eine bessere Übersicht über die aktuelle Autonomie des Roboters zu gewähren.

Der erste Grad erlaubt die Steuerung des Roboters mit Hilfe intuitiver Bewegungskonzepte. Die Autonomie des Roboters ist in dieser Stufe minimal und dient ausschließlich zur Vermeidung von Gefahren. Jede Bewegung und Bewegungsänderung wird durch die Interaktion des Anwenders eingeleitet. Gesteuert werden kann die Blickrichtung, der omnidirektionale Antrieb und die Position des Endeffektors samt Öffnungswinkel der Hand.

Im zweiten Autonomiegrad, dem halb-autonomen Modus wurden vier Aktionsprimitive implementiert. Es handelt sich um häufig ausgeführte Aktivitäten, die der Roboter autonom ausführen soll. Obwohl es sich um primitive Aktionen handelt, wie die autonome Navigation zu einem Ort oder das Greifen eines Gegenstandes, müsste ein Anwender für diese Handlungen viel Zeit, Aufmerksamkeit und Geschicklichkeit opfern. Die Interaktion des Anwenders beschränkt sich auf dieser Stufe auf einen einzelnen Klick, der Wahl einer Position oder die Auswahl eines Objektes.

Im höchsten Autonomiegrad werden semantische Ziele genutzt, um eine Aufgabe vollständig zu definieren. Der Roboter hat auf dieser Ebene eine Vorstellungen von Orten, Gegenständen und möglichen Aufgaben. Diese können auf Wunsch vollständig autonom ausgeführt werden.

4.4 Erste Autonomiestufe

Durch die erste Autonomiestufe besitzt der Anwender die Möglichkeit mit Hilfe des Roboters die Umwelt zu verändern und Handlungen auszuführen.
Es wurde ein Ansatz der abgesicherten Teleoperation gewählt, bei welcher der Anwender die volle Kontrolle über den Roboter besitzt. Lediglich in Gefahrensituationen greift der Roboter ein und überschreibt die Steueranweisungen des Anwenders. Mit Hilfe dieser Methode sind Verzögerungen in der Kommunikation weniger ge-

fährlich. Die Stufe ermöglicht Handlungen, die in dieser Form auf höheren Autonomiestufen nicht angeboten, berücksichtigt oder sogar verboten sind. Das Aufheben dieser Beschränkungen erlauben dem Anwender einmalige Aktivitäten auszuführen, welche einen zu speziellen Charakter besitzen, um vom Roboter autonom gelöst zu werden.
Die Steuerung ist so konzipiert, dass eine fehlerhafte Wahrnehmung der Umwelt den Anwender möglichst wenig beeinträchtigt. Werden Annahmen vorausgesetzt, so hat der Anwender jederzeit die Möglichkeit diese zu widerrufen, um eine gewünschte Aktivität ungehindert ausüben zu können. Die Autonomie nimmt in dieser Stufe deshalb eine passive Rolle ein.
Mit Hilfe eines übertragenen Kamerabildes erhält der Anwender Einblick in die entfernte Umgebung.

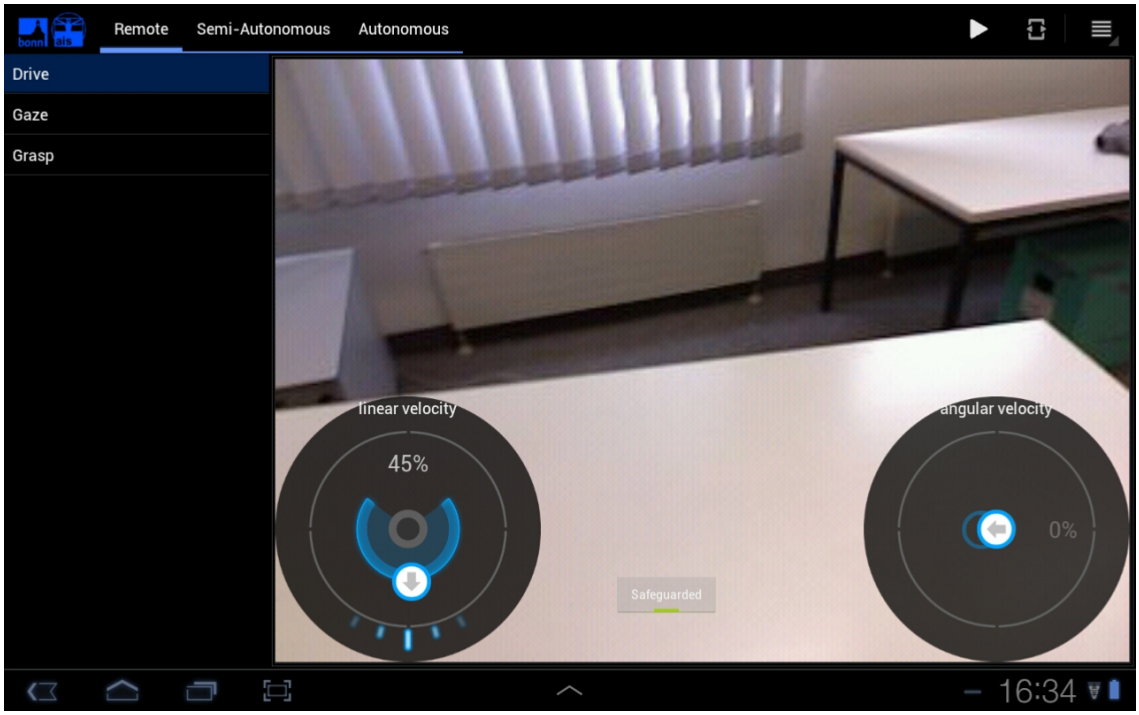


Abbildung 10: Steuerung der Basis durch den Einsatz von Joysticks

Fortbewegung Die Ansicht zur Steuerung des omnidirektionalen Antriebs besteht aus einem bildschirmfüllenden Kamerabild, zwei virtuellen Joysticks und einem Schalter zur Deaktivierung der Kollisionsvermeidung (siehe Abbildung 10).
Der Einsatz von zwei Joysticks bietet sich an, um den vollen Funktionsumfang des Antriebs auszuschöpfen. Der Anwender erhält die Möglichkeit den Roboter auf der Stelle zu drehen, Kurven zu fahren oder den Roboter seitlich fortzubewegen.
Die Funktion der abgesicherten Steuerung liefert der lokale Planer des NimbRo-Navigation Pakets. Er sorgt für eine Kollisionsvermeidung. Berücksichtigt werden dabei die Abstandsmessungen des in der Basis integrierten Laserscanners. Der Öffnungswinkel dieses Lasers beträgt 270 Grad. Der Roboter besitzt deshalb nur

ein eingeschränktes Sichtfeld, in welchem Hindernisse erkannt werden. Steuert der Anwender auf ein Hindernis zu, so sorgt der lokale Planer für ein Abbremsen des Roboters bis hin zu einem Stillstand, bevor er mit dem Objekt kollidiert. Da die Autonomie keine aktive Rolle einnehmen soll, wird ein Hindernis in dieser Stufe nicht umfahren, sondern ausschließlich erkannt und eine Kollision verhindert und gemeldet.
Keine Beachtung finden bei diesem Ansatz Hindernisse, die auf Grund ihrer Lage vom Laser nicht erfasst werden. Abhilfe schafft der Einsatz eines erweiterten lokalen Planers (`local_motion_planer_3d`), welcher die Daten von mehreren Laserscannern verarbeitet.
Auch das Tiefenbild der Kamera lässt sich in ein passendes Format wandeln und verwenden. Als problematisch erweist sich jedoch die große Dynamik durch die Vielzahl der Freiheitsgrade bis hin zur RGB-D-Kamera. Auf den Einsatz des Tiefenbildes als kontinuierliche Quelle für den lokalen Planer wird deshalb verzichtet.
Durch den Einsatz mehrerer Laserscanner werden somit auch Tische und andere Hindernisse erfasst. Im herkömmlichen vollständig autonomen Betrieb des Roboters sind Tische in einer Karte als belegte Zelle eingezeichnet. Der zur Navigation eingesetzte globale Planer sorgt mit dieser Information dafür, dass kein Pfad durch oder in eine Tischplatte geplant wird.
Auf den Einsatz einer Karte wird auf dieser Autonomiestufe verzichtet, da sie eine ausreichende Lokalisierung des Roboters voraussetzt. Ist die Lokalisierung schlecht oder nicht vorhanden, wird der Betrieb unnötig gestört.

Der Versuch rückwärts zu Fahren wird im abgesicherten Modus unterbunden. Der Roboter besitzt keinen Einblick in die rückwärtige Richtung. Es muss deshalb davon ausgegangen werden, dass sich ein unmittelbares Hindernis hinter ihm befindet.
Wird ein Roboter aus der Ferne unterstützt durch das Kamerabild gesteuert, so hat der Anwender automatisch den Drang den Roboter in die gewünschte Richtung zu drehen bevor die Fahrt aufgenommen wird. Diese Einschränkung bleibt dem Anwender deshalb häufig verborgen.

Hat der Anwender den Eindruck, dass der Roboter auf Grund einer falschen Wahrnehmung eine Bewegung verweigert oder soll gezielt eine Aktivität unternommen werden, die der Roboter vermeidet, so besteht die Möglichkeit die Kollisionsvermeidung zu deaktivieren. Damit wird die unmittelbare Steuerung der Basis erlaubt.
Der lokale Planer muss für diesen Zweck modifiziert werden. Das ursprüngliche Verhalten sendet in einem regelmäßigen Intervall eine Stopp-Nachricht an die Basis, wenn der Planer untätig ist. Je nach Häufigkeit unterbinden oder stören diese Nachrichten eine direkte Steuerung. Ursprünglich wurde dieses Verhalten gewählt damit die Basis nicht versehentlich das zuletzt empfangene Kommando weiter ausübt und aus diesem Grund nie zum Stillstand kommt. Wird das kontinuierliche Senden von Stopp-Nachrichten unterbunden, ist zu beobachten, dass die Basis

wesentlich später zum Stillstand kommt. Eine Schutzschaltung sorgt dabei für das Stehenbleiben des Roboters, wenn neue Steueranweisungen ausbleiben. Der lokale Planer wurde so angepasst, dass er nach der Fertigstellung seiner Aufgabe eine einzelne Stopp-Nachricht veröffentlicht und die Basis nicht weiter beeinflusst. Dies ermöglicht das überwinden von Engpässen, das Aufschieben einer angelehnten Türe oder das Durchqueren von vermeintlichen Hindernissen.

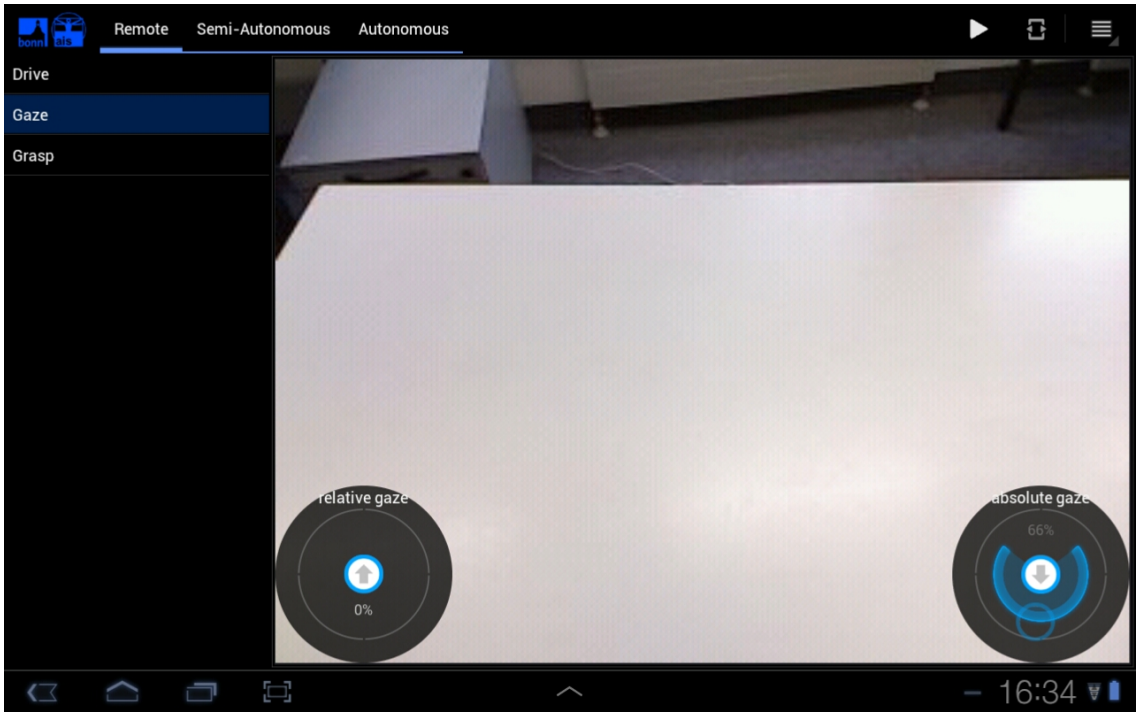


Abbildung 11: Steuerung des Blicks

Blick Über den Menüeintrag „Gaze“ ist der Kopf und damit die Blickrichtung des Roboters steuerbar. In dieser Ansicht wird ebenfalls ein bildschirmfüllendes Kamerabild eingesetzt, welches mit Hilfe von zwei virtuellen Joysticks (siehe Abbildung 11) oder dem Einsatz von Gesten manipuliert werden kann.
Der Kopf und die daran montierte Kamera ist über zwei Gelenke schwenk- und neigbar. Einen optischen Zoom bietet die eingesetzte Kamera nicht.
An dieser Stelle kommt zum ersten Mal das empfohlene Prinzip von Olsen und Goodrich [GO03] zum Einsatz, die Umwelt zu manipulieren und nicht den Roboter. Durch die Verwendung von einfachen Gesten entsteht der Eindruck als wäre die präsentierte Welt manipulierbar. Das Wischen von links nach rechts hat ein Ziehen des Bildes in die gleiche Richtung zur Folge. Der Roboterkopf wird dazu jedoch nach links gedreht, um den Blick auf das in die Bildschirmmitte gezogene Bildsegment zu erlauben.

Zum Vergleich werden zwei virtuelle Joysticks eingesetzt, welche die Gelenkstellungen des Roboters ohne einer zur Verfügung stehenden Metapher steuern.

Der linke virtuelle Joysticks dient zur absoluten Steuerung der Blickrichtung. Jede Bewegung des Joysticks hat eine temporäre Auslenkung des Kopfes zur Folge. Durch das Zurückkehren zur Nullposition kehrt auch der Kopf zurück in die initiale Ausrichtung. Der zweite Joystick bewirkt eine relative Änderung der Blickrichtung und erlaubt das langfristige Anvisieren eines Ortes von Interesse.

Die bisherige Funktionalität des Roboters sieht zwei Modi zur Steuerung des Kopfes vor. Zum einen die automatische Ausrichtung in eine Fahrtrichtung in Abhängigkeit von der aktuellen Geschwindigkeit und zum anderen der Blick auf eine Pose im Raum.
Der erste Modus wird bei vollständiger Autonomie eingesetzt, um dem Roboter ein annähernd menschliches Verhalten zu verleihen. Er vermittelt den Eindruck, als würde der Roboter mit Hilfe seiner Augen die Wegstrecke kontrollieren. In Wirklichkeit reicht die Neigung des Kopfes jedoch nicht aus, um mit der leicht gekippten Kamera einen optimalen Einblick während der Navigation zu erhalten. Die unruhige und oft hektische Kopfbewegung wird vom Anwender auch auf Grund der leichten Verzögerung des Kamerabildes als störend empfunden. Erweitert wurde die zentrale Steuereinheit des Roboters (`robot_control`), um die Ansteuerung des Kopfes durch eine kontinuierliche Richtungsgeschwindigkeit und der Wahl einer absoluten Winkelstellung in Grad.

Im Verlauf der Tests hat sich herausgestellt, dass ein virtueller Horizont, beziehungsweise eine Information über die aktuelle Drehung des Kopfes, vorteilhaft ist. Ohne diese Information ist der Anwender in der Pflicht, sich die aktuelle Orientierung des Kopfes zu merken, um nicht versehentlich die Basis in eine falsche Richtung zu steuern. Verbunden mit dieser Information ließe sich eine Neuausrichtung ausführen. Dabei wird auf Wunsch die Basis in die Richtung des Kopfes gedreht und der Kopf in die entgegengesetzte Richtung, so dass beide wieder eine Linie bilden. Der Ansatz die Richtung des Kopfes als Orientierung des Roboters zu wählen fördert neue Probleme zu Tage, wie die ungünstige Lage des Oberkörpers zu einem Objekt und das schnelle Erreichen von Limitierungen bei der Drehung des Oberkörpers.

Greifen Der letzte Menüpunkt trägt die Bezeichnung „Grasp“ und erlaubt die Steuerung der sechs Freiheitsgrade des Roboterarms. Um dem Anwender die Kontrolle zu erleichtern ist die Endeffektorpose mit Hilfe von vier Joysticks steuerbar. Zwei der Joysticks sind dabei nur mit jeweils einem Freiheitsgrad belegt (siehe Abbildung 12). Die vom Roboter zur Verfügung gestellte inverse Kinematik sorgt für die resultierende Gelenkstellung des Arms.

Ein spezieller entworfener Knoten namens `nimbro_teleop_arm` fusioniert die Steueranweisungen der Joysticks und den Öffnungswinkel der Hand. Die vollständige Endeffektorpose wird als Bewegungsprimitiv an die inverse Kinematik übergeben.
Es existiert kein Schutzmechanismus, der dazu beiträgt Kollisionen mit der Umwelt

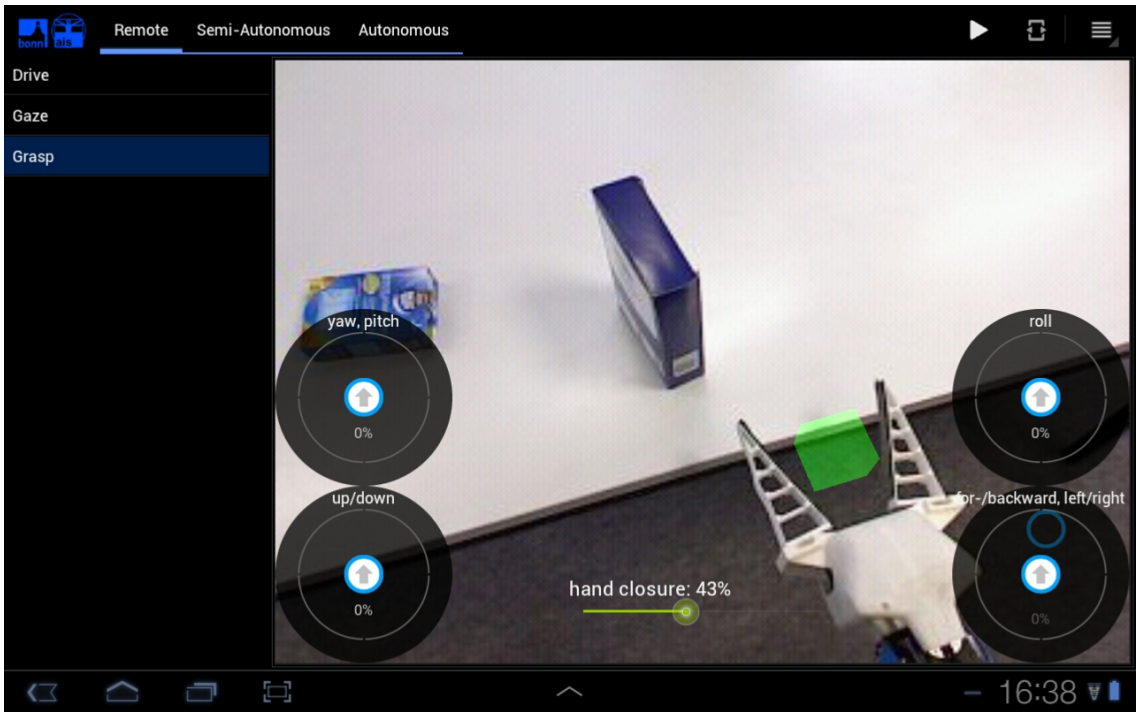
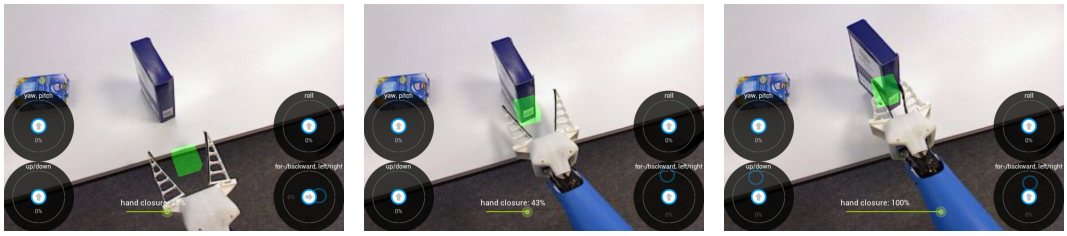


Abbildung 12: Manuelles Greifen eines Gegenstandes



(a) Öffnen der Hand (b) Zielanfahrt (c) Anheben des Gegenstandes

Abbildung 13: Illustration der manuellen Steuerung des Arms

oder dem Roboter zu vermeiden. Im Idealfall sollte an dieser Stelle ebenfalls eine abgesicherte Steuerung zum Einsatz kommen, um das Gefahrenpotential zu reduzieren. Dies kann darüber hinaus komplizierte Greifszenarien ermöglichen, wenn die Kollisionsvermeidung ausschließlich die Berührung von Objekte mit der Hand des Roboters zulässt. Unter- und Oberarm sollten dabei die Berührung von Hindernissen vermeiden.

Um Fehler bei der Bedienung zu entschärfen und unkontrollierte Bewegungen des Armes zu vermeiden wurde die Geschwindigkeit, mit der das gewünschte Ziel angefahren wird auf 2.5 Zentimeter pro Sekunde herabgesetzt.
Die geringe Geschwindigkeit vermeidet außerdem, dass der Arm in Schwingung gerät und eine ruckartige Steuerung den Betrieb erschwert. Um trotzdem eine zügige und reibungslose Steuerung der Endeffektorpose zu gewährleisten, wird diese

perspektivisch korrekt in das übertragene Kamerabild projiziert. Der Anwender erhält damit die Möglichkeit die Zielposition der Hand zu kontrollieren bevor der Roboter diese erreicht hat.
Steuert der Anwender den Arm langsam und vorsichtig, so bewegen sich Hand und Projektion simultan (siehe Abbildung 17). Dieser Ansatz hilft über Probleme bei auftretenden Verzögerungen hinweg und ist weniger anfällig bei unzureichender Wahrnehmung der Umwelt.

Für den Einzugsbereich des Armes ist ein künstlicher Korridor vor den Schultern des Roboters definiert. Dies verhindert die Steuerung des Armes zu Posen, die außerhalb seiner Reichweite liegen. Auch der unnatürliche und schwer kontrollierbare Griff hinter den Roboter wird damit unterbunden.

Diese direkte Steuerung des Armes eröffnet dem Anwender eine Vielfalt an Möglichkeiten. Er ist in der Lage Aktionen auszuführen, welche auf höheren Autonomiestufen nicht möglich sind. Dabei kann es sich um eine komplizierte Greifbewegung handeln, als auch um das Aufstoßen einer Tür, die Bedienung der Mischbatterie eines Wasserhahns und vieles mehr.

4.5 Zweite Autonomiestufe

In der zweiten Autonomiestufe, als „Semi-Autonomie“ bezeichnet, wird das Konzept der geteilten Steuerung eingesetzt. Der Anwender interagiert hierbei mit dem Roboter, indem er eine Arbeitsanweisung erteilt. Diese Arbeitsanweisung wird vollständig autonom ausgeführt.
Murphy [Mur00] beschreibt diesen Ansatz bildlich am Beispiel von Eltern, die ihrem zehnjährigen Sohn einen Arbeitsauftrag geben. Die Eltern besitzen dabei das Wissen, dass der Sohn eine Aufgabe, wie das Aufräumen seines Zimmers selbstständig erledigen kann. Sie kümmern sich jedoch nicht um Details, wie oder in welcher Reihenfolge der Raum aufgeräumt wird.
Dieser Steuerungsansatz bietet sich an, um den Anwender zu entlasten. Er ist bekannt aus Anwendungen, in denen ein Operator mehrere Roboter steuert. Nach der Erteilung eines Arbeitsauftrages besteht keine Notwendigkeit den Roboter bei der Ausführung seiner Arbeit zu überwachen. Der Anwender kann sich deshalb auf den nächsten Roboter und dem Erteilen der nächsten Aufgabe konzentrieren.

Die in diesem Autonomiegrad angebotenen Aktionen erfüllen Aufgaben, die ein Dienstleistungsroboter regelmäßig bewerkstelligen muss. Diese häufig wiederholten Abläufe werden als Aktionsprimitive bezeichnet. Es handelt sich um primitive Aktivitäten, die trotz der augenscheinlich geringen Komplexität vom Anwender ein hohes Maß an Konzentration erfordern. Umso häufiger ein Anwender diese Handlungen ausführt, desto nachlässiger wird er und die Wahrscheinlichkeit eines Bedienfehlers steigt.
Die Bedienung erfolgt in dieser Stufe nicht mit Hilfe von kontinuierlichen Steuerbefehlen, sondern durch die Erteilung eines einzelnen Auftrags. Latenz

und Bandbreite sind in diesem Modell weniger kritisch. Es muss ausschließlich eine fehlerfreie Übermittlung des Arbeitsauftrages an den Roboter gewährleistet sein. Eine weitere Anforderung an den Roboter ist das robuste Ausführen des Aktionsprimitives.

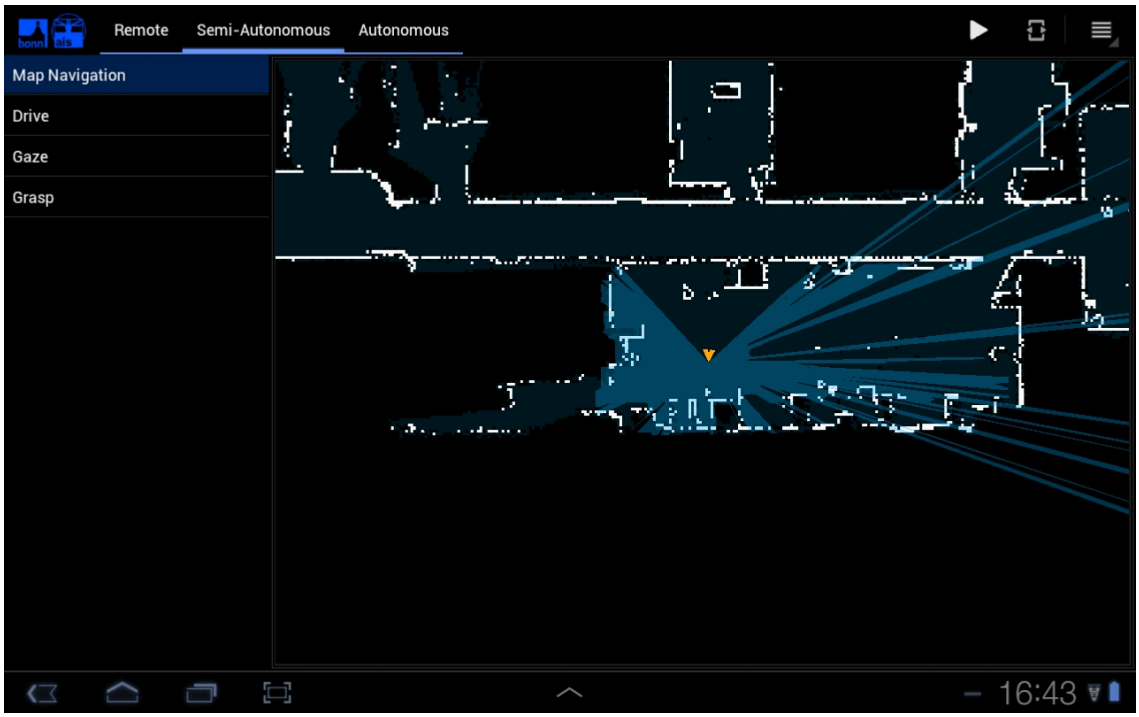


Abbildung 14: Navigation auf einer globalen Karte

Navigation auf einer Karte Ein für den Anwender anspruchsvolles Aktionsprimitiv ist die Überbrückung von größeren Distanzen mit dem Roboter.
Möchte man einen Roboter in die Küche schicken, um einen Getränk zu holen bietet sich die Navigation auf einer Karte an. Für diese Aufgabe kommt eine vorgefertigte Belegungskarte der Umgebung zum Einsatz. Auf dieser wird geplant und navigiert (siehe Abbildung 14).
Ein erweiterter Monte-Carlo Algorithmus (`amcl`) sorgt mit Hilfe eines Laserscanners für die Lokalisation des Roboters in der Karte. Bedient wird sich der Annahme, dass die Umgebung, in welcher der Dienstleistungsroboter arbeitet nur marginal verändert wird. Notwendig für den Algorithmus und die resultierende Lokalisation ist eine initiale Schätzung. Auf diese Weise besitzt und pflegt der Roboter eine Information über seine aktuelle Lage in der Karte.
Der globale Planer (`global_planner`), ebenfalls ein Bestandteil des NimbRo-Stacks, sorgt für die Berechnung eines Pfades zwischen Ziel und Roboter. Dieser Pfad besteht aus einer Reihe aufeinander folgender Streckenabschnitte, welche Punkt für Punkt an den lokalen Planer übermittelt werden. Dynamische Hindernisse stellen somit keine unmittelbare Gefahr dar, obwohl der globale Planer zum Zeitpunkt der Planung keine Kenntnis von ihnen besitzt. Sie werden, sollte der

direkte Weg versperrt sein, umfahren. Meldet der lokale Planer, dass ein Hindernis wie eine geschlossene Tür nicht umfahren werden kann, wird eine Neuplanung vorgenommen.
Um eine Visualisierung des geplanten Pfades in der Benutzeroberfläche zu erlauben wurde der globale Planer erweitert. Er veröffentlicht nach erfolgreicher Berechnung die vollständige Route. Wird ein Zwischenziel erreicht, erfolgt eine Aktualisierung des Pfades.

Die vom Roboter genutzte Belegungskarte wird komprimiert an das Endgerät übertragen und dem Benutzer dargestellt. Zusammen mit der aktuellen Pose des Roboters und einem eingezeichneten Laserscans kann sich der Anwender einen Überblick verschaffen.
Durch einen Klick in die Karte kann der Anwender ein Fahrtziel definieren. Durch Ziehen des Fingers lässt sich die Orientierung der Pose definieren. Nach erfolgreicher Planung wird der Pfad in die Karte gezeichnet.
Mit einfachen Gesten besitzt der Anwender die Möglichkeit die Karte zu verschieben oder die Skalierung zu verändern. Der visualisierte Laserscan erlaubt dem geübten Benutzer die Güte der Lokalisation zu beurteilen, die Ursache für eine Neuplanung oder ein Stehen bleiben zu identifizieren.

Das gemeinsame Koordinatensystem, in welchem die Daten dargestellt werden trägt die Bezeichnung „*map*“.
Daten aus anderen Koordinatensystemen wie die Pose des Roboters oder dem Laserscan werden in dieses Koordinatensystem von der Android-Anwendung transformiert. Dazu werden Transformationen vom `tf`-Knoten angefordert. Diese Art der Berechnung von Transformationen ist gängige Praxis und bietet eine große Arbeitserleichterung. Das Datenaufkommen zu diesem Thema kann jedoch enorme Ausmaße annehmen, da ROS keine Möglichkeit bietet bestimmte Transformationen von Interesse zu selektieren. Es werden alle verfügbaren Transformationen in einem zehn oder zwanzig Herz Intervall übertragen. In zukünftigen ROS-Versionen wird sich durch die Einführung von `tf2` diesem Problem angenommen. Als Übergangslösung bietet sich eine Vorverarbeitung der entsprechenden Nachrichten auf dem Roboter an. Befinden sich alle übertragenen Daten im gleichen Koordinatensystem, so kann auf eine Transformation verzichtet werden. Eine Beispiel für einen derartigen Knoten findet sich im *rosjava*-Stack mit der Bezeichnung `tf_frame_map_pose_publisher`.

Fortbewegung In einem weiteren Aktionsprimitiv bestimmt der Anwender durch einen Klick in ein übertragenes Kamerabild ein Fahrtziel (siehe Abbildung 15). Dieses befindet sich in Sichtweite des Roboters und wird autonom angefahren.
Um die Anwendung und die drahtlose Verbindung zu entlasten, wird ausschließlich die berührte Bildkoordinate an den Roboter übertragen. Ein für diesen Zweck entwickelter Dienst nimmt die Koordinaten entgegen und transformiert sie mit Hilfe des registrierten Tiefenbildes der RGB-D-Kamera in eine Zielpose relativ zum Roboter. Da nicht für jede Bildkoordinate eine Tiefeninformation zur Verfügung steht wird

gegebenenfalls über umliegende gemittelt.
Wurde die Zielpose erfolgreich bestimmt, wird diese in Form eines Markers veröffentlicht und im Kamerabild der Anwendung eingezeichnet. Gleichzeitig wird der lokale Planer aufgefordert dieses Ziel im roboter-eigenen egozentrischen Koordinatensystem anzufahren. Dabei wird der Planer nicht mit einer Richtungsgeschwindigkeit versorgt, wie es im ersten Autonomiegrad der Fall ist, sondern mit einer Zielpose. Dies sorgt für die Umfahrung von auf dem Weg liegender Hindernisse mit dem informierten Suchalgorithmus A-Stern.
Durch den ausschließlichen Einsatz des lokalen Planers ist keine Lokalisierung des Roboters in seiner Umgebung notwendig. Wird eine Zielpose gewählt, welche der Roboter nicht erreichen kann, so wird diese so nah wie möglich angefahren. Dies ist hilfreich bei der Anfahrt eines Tisches. Die Information, dass kein Pfad zum Ziel geplant werden kann, würde einen Anwender verärgern und der Lösung eines Problems nicht näher bringen.

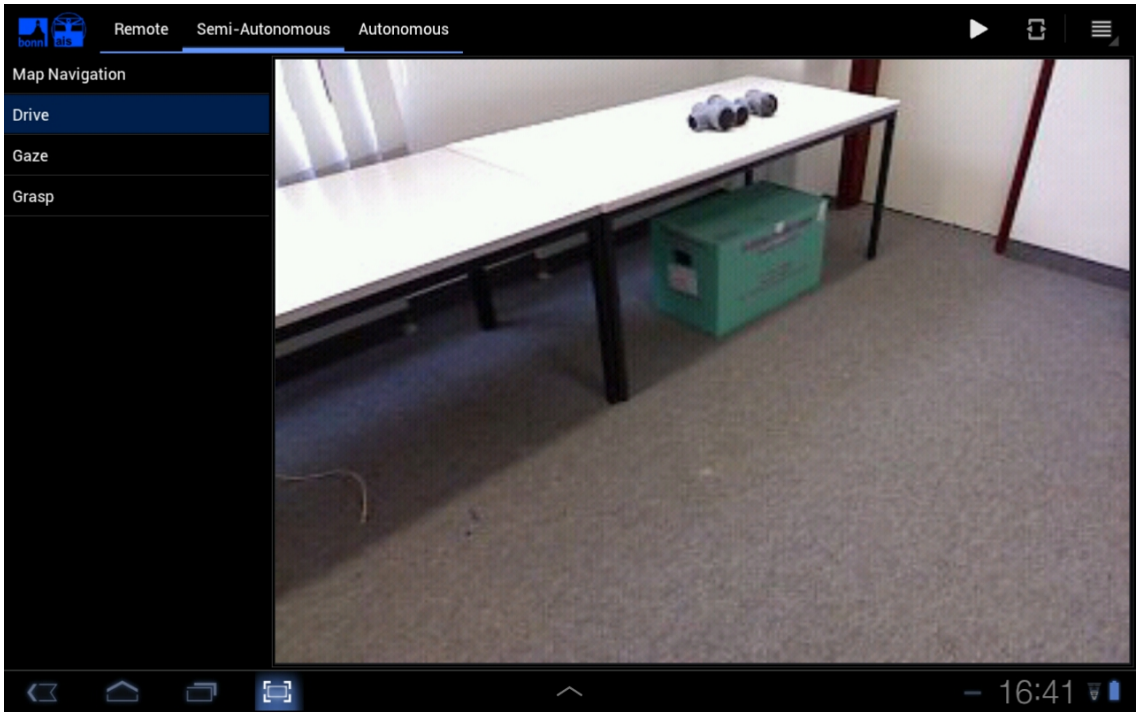


Abbildung 15: Navigation zu einer Pose im Kamerabild

Das Ziel ist die weitgehende Entlastung des Anwenders. Es ist keine Vorstellung von den Besonderheiten eines omnidirektionalen Antriebs notwendig. Der Roboter übernimmt für den gesamten Bearbeitungszeitraum die Hoheit über die Steuerung der Basis. Die Repräsentation der Szene ist über das Kamerabild so natürlich wie möglich gewählt.

Blick Das Aktionsprimitiv zur Steuerung des Blicks ist nahezu identisch mit der zuvor beschriebenen Fortbewegung.

Eine Berührung des Kamerabildes sorgt dabei für die Ausrichtung des Kopfes auf das gewünschte Ziel. Es kommt wiederum der eigenes entwickelte Knoten `pcl_to_pose` zum Einsatz. Über einen zusätzlichen Parameter beim Aufruf des Dienstes wird nach der Berechnung der Zielpose die zentrale Steuereinheit `robot_control` beauftragt den Kopf zu drehen.

Konzeptionell erscheint die Einordnung dieser Steuerung auf der halb-autonomen Stufe als sinnvoll. In der Praxis bringt eine strikte Trennung jedoch viele Nachteile mit sich. Der Anwender hat häufig den Drang eine initiale Position durch einen Doppelklick zu bestimmen. Weicht diese Blickrichtung von dem gewünschten Ziel ab, so wird kein erneuter Versuch unternommen das Ziel über eine Berührung zu setzen. Stattdessen versucht der Anwender mit Hilfe einer Wischbewegung das Ergebnis anzupassen.

Ursache für den geringen Mehrwert der halb-autonomen Blicksteuerung ist die Gefahrlosigkeit und der unerheblich geringere Zeitaufwand. Das Anvisieren eines Ziels mit Hilfe von Gesten beansprucht den Benutzer nur wenige Sekunden. Die Bewegung der beiden Freiheitsgrade sind schnell verständlich und leicht zu erlernen. Eine falsche Bedienung wird schnell erkannt und ist ungefährlich.
Bei der Wahl eines Ziels per Klick wartet der Benutzer auf das gewünschte Ergebnis, um seine Aktion zu kontrollieren. Gegebenenfalls wird das Ergebnis korrigiert.
Der Anwender wird das Endgerät nicht aus der Hand legen, da das Ergebnis unmittelbar bevorsteht. Eine Unterscheidung zwischen Interaktionszeit und Bearbeitungszeit sollte deshalb nicht unternommen werden. Die Operation ist zu kompakt, um einen signifikanten Mehrwert aufzeigen zu können.

Möglicherweise ist der Einsatz des Tiefenbildes in diesem Zusammenhang übertrieben und eine einfache geometrische Berechnung der Auslenkung des Kopfes auf der ersten Autonomiestufe ist ausreichend und unanfälliger gegenüber Messfehlern.
Interessanter wird das beschriebene Aktionsprimitiv, sobald die Entfernung zu einem Ziel eine Rolle spielt. Auf diese Weise lässt sich ein optischer Zoom kontrolliert steuern.
Ein weiterer für den Benutzer entlastender Ansatz ist das kontinuierliche Fixieren eines Ziels. Dabei wird der Blick des Roboters an eine Pose geheftet, welche während der Navigation im Blick gehalten wird. Dieses Problem kann sich bis hin zur Orientierung der Basis erstrecken, um sowohl eine sichere Fortbewegung zu gewährleisten als auch eine Fixierung der Pose.

Greifen Das Ziel des halb-autnomen Greifens von Objekten ist den Anwender vergleichbar wenig zu belasten, wie bei der Navigation zu einem Ort. Dabei führt der Roboter den vollständigen Greifvorgang autonom aus. Der Anwender besitzt ausschließlich die Aufgabe ein als greifbar markiertes Objekt auszuwählen (siehe Abbildung 17).

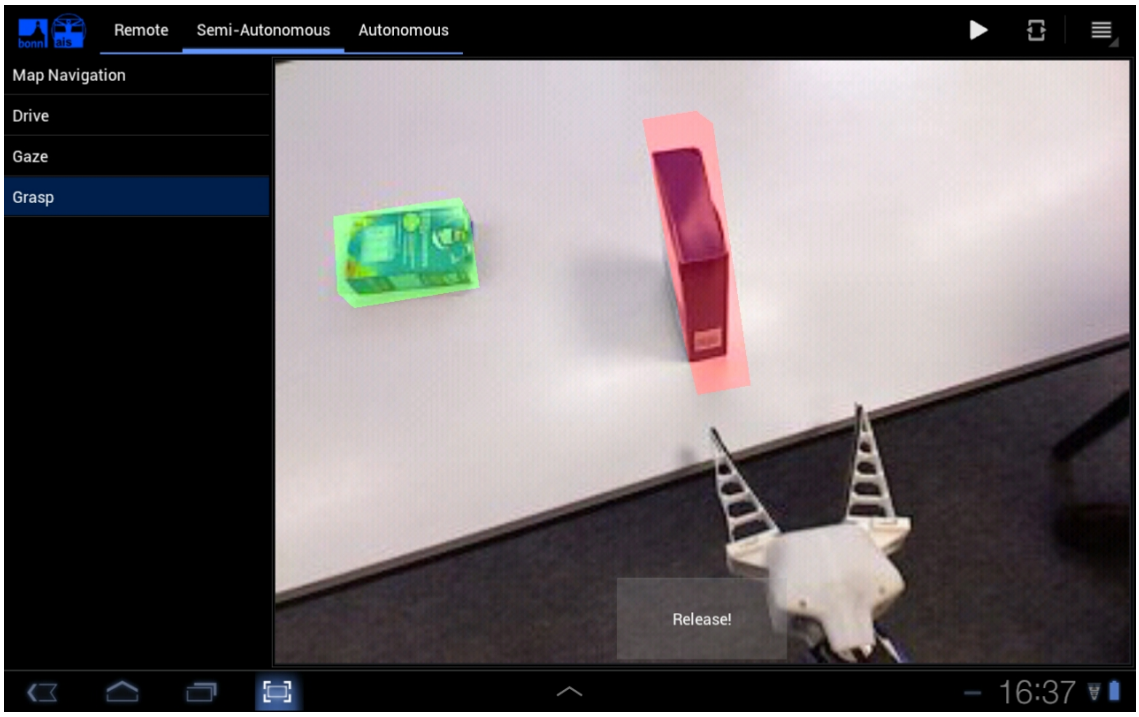
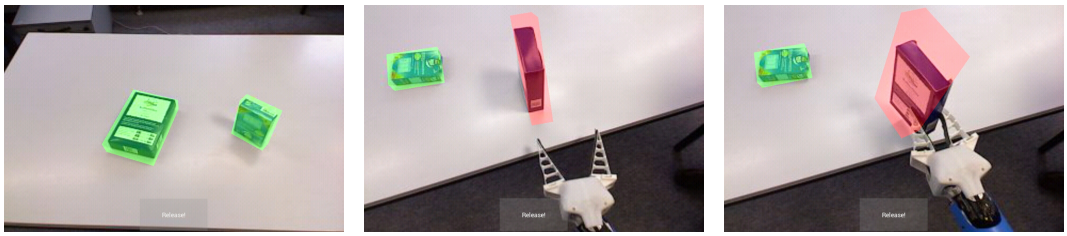


Abbildung 16: Greifen erkannter Gegenstände



(a) (b) (c)

Abbildung 17: Illustration des halbautonomen Greifens

Die Auswahl erfolgt innerhalb des übertragenen Kamerabildes des Roboters. Objekte werden im Kamerabild farblich herausgehoben (siehe Abbildung 16). Der Anwender kann ein solches Objekt über eine Berührung auswählen. Das gewählte Objekt wird rot eingefärbt und der Greifvorgang eingeleitet. Trotz der einfachen Interaktion sind eine Vielzahl von Komponenten für die Bearbeitung dieser Aufgabe notwendig.

Für die Bestimmung der greifbaren Objekte wird die sogenannte `segmentation_node` aus dem Paket `nimbro_kinect` eingesetzt.
Der Knoten arbeitet mit der Annahme, dass Objekte auf einer gemeinsamen Ebene, wie einer Tischplatte, stehen. Untersucht wird dabei das Tiefenbild der RGB-D-Kamera. Sind die Mindestanforderungen für eine zusammenhängende Ebene erfüllt, so wird diese auf Objekte untersucht, die aus ihr heraus stehen. Die

gefundenen Objekte samt geschätzter Ausmaße und Orientierung werden an den `multi_object_tracker` übergeben. Dieser hat die Aufgabe die Schätzung der Position von Objekten über einen längeren Zeitraum zu verwalten.
Eingesetzt wird ein Kalman-Filter, der neue Messungen bekannten Objekten zuordnet oder neue Objekte im Modell anlegt. Gehen keine Messungen für ein Objekt mehr ein, so sinkt die Relevanz bis zu einer Entfernung aus dem System.

Jedes vom Tracker gepflegte Objekt erhält eine eindeutige Identifikationsnummer. Diese wird eingesetzt für die Auswahl eines Objektes in der grafischen Oberfläche und den anschließenden Greifvorgang des Roboters.
Zur Visualisierung werden die gefundenen Objekte als Bounding Volumen in das Kamerabild projiziert. Bei den Bounding Volumen handelt es sich um an den Achsen des Objektes ausgerichtete umschließende Quader. Auf diese Weise bleibt die Orientierung und der Eindruck eines dreidimensionalen Objektes erhalten.

Für den Greifvorgang wurde die bestehende Komponente `mobilemanipulation_fsm` erweitert. Im ursprünglichen Zustandsautomaten wird mit Hilfe eines vertikal gedrehten Lasers am Torso des Roboters die Höhe einer Tischkante bestimmt. Ist diese bestimmt, wird der Laser horizontal gedreht und am Torso des Roboters leicht über die Ebene des Tisches gefahren. Auf diese Weise ist es möglich mit dem Laserscanner und einer `laser_object_detection` Objekte zu erkennen, die sich auf dem Tisch befinden. Erst ab diesem Zeitpunkt wird im ursprünglichen Zustandsautomaten der Objekt-Tracker aktiviert.
Befindet sich kein Objekt in Reichweite des Arms, so wird sich am nächstgelegenen Objekt neu ausgerichtet. Auch eine Objektidentifizierung kann eingesetzt werden. Dabei fokusiert der Roboter jedes einzelne verfolgte Objekt und versucht es mit Hilfe seines Wissens zu identifizieren. Wurde das gesuchte Objekt erkannt wird der Tracker deaktiviert und wiederum eine Neuausrichtung der Basis ausgeführt. Gegriffen wird in jedem Fall das zum Roboter nächstgelegene Objekt.

Dieser Zustandsautomat wurde angepasst, um eine frühzeitige Erkennung von Objekten und dem Anwender eine gezielte Auswahl zu erlauben.
Zur Objekterkennung die RGB-D-Kamera in Kombination mit der zuvor beschriebenen Segmentierung eingesetzt. Die Position der erkannten Objekte wird mit Hilfe des Trackers gepflegt. Ziel ist das langfristige Halten der Identifikationsnummer eines Objektes. Geht die Zuordnung verloren, nachdem das Objekt ausgewählt wurde, ist die zuverlässige Lösung der Aufgabe nicht mehr gewährleistet. Der Tracker läuft deshalb während des gesamten Vorgangs. Die Toleranz des Kalman-Filters wurde angepasst, um eine leichte Bewegung der Objekt zu erlauben. Die Beziehung zwischen Laser und Objekt war bisher starr. Während der kurzen Betriebszeit des Trackers wurde das Modell nicht bewegt. Durch den Einsatz der Kamera zur Objekterkennung ist das Modell nun einer starken Dynamik ausgesetzt. Zuordnungen sollen gehalten werden, auch während der Neuausrichtung an einem Objekt oder der Bewegung des Kopfes.

Wurde ein Objekt ausgewählt, so wird dieses dauerhaft anvisiert, um es nicht aus den Augen zu verlieren.
Nach einer Ausrichtung an diesem Objekt wird die Greifpose an Hand der Identifikationsnummer des Objektes bestimmt. Ist während der Ausrichtung die Zuordnung verloren gegangen, wird das nächstgelegene Objekt gegriffen.
Das Halten der Zuordnungen hat zwei Vorteile: Der Anwender kann auch nach der Auswahl eines Objektes erkennen, welches Objekt von ihm gewählt wurde und es wird verhindert, dass der Roboter auf Grund eines Navigationsfehlers oder einer Ungenauigkeit einen falschen Gegenstand greift.

Der Laserscanner im Torso wird nur noch zur Bestimmung der Tischhöhe eingesetzt. Für weitere Aufgaben müssen viele für den Anwender unverständliche Randbedingungen erfüllt werden. Der Laser kann erst Objekte erkennen, wenn er sich oberhalb einer Tischplatte befindet. Der Anwender müsste also mit einer unnatürlichen Aufgabe, wie dem Suchen einer Tischplatte beauftragt werden bevor Objekte angeboten werden können. Werden Objekte erkannt, so kann durch den Laser nur die Breite bestimmt werden. Für die Visualisierung ist diese Information mangelhaft. Objekte können lediglich mit einem Band versehen werden. Eine natürliche Darstellung erweist sich als schwierig.

4.6 Dritte Autonomiestufe

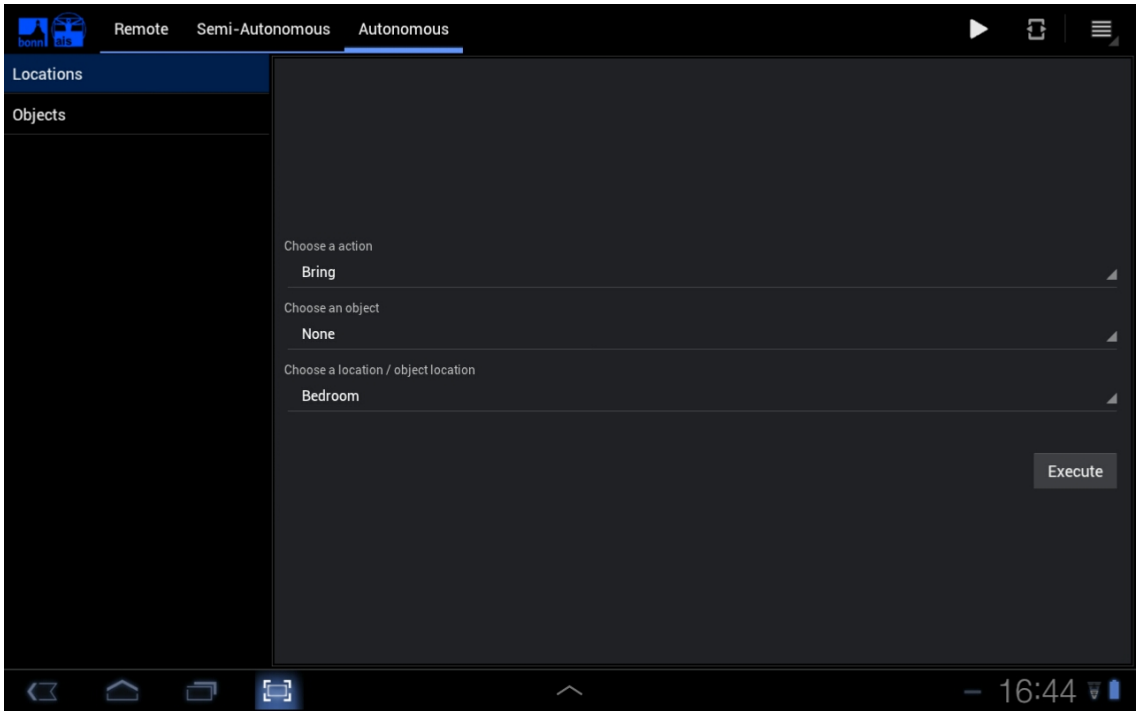


Abbildung 18: Dialogsystem der vollständigen Autonomie

In der dritten Autonomiestufe, der vollständigen Autonomie kann ein umfassender Arbeitsauftrag erteilt werden. Das notwendige Wissen stellt der Roboter bereit. Er besitzt Kenntnis über die Bezeichnung von Orten, ist in der Lage Objekte zu identifizieren und hat eine Ahnung von zur Verfügung stehender Aktivitäten. Zum Einsatz kommt ein ROS-Knoten aus dem Nimbro-Stack, der einen bestehenden Zustandsautomaten zur Verfügung stellt. Er wurde entwickelt zur Lösung einer Testaufgabe der Robocup@Home-Liga namens GPSR (General purpose service robotic). Im ursprünglichen Anwendungsfall werden Aufträge mündlich an den Roboter übermittelt. Auf dem Roboter kommt eine Spracherkennung zum Einsatz, welche einen gehörten Satz mit einer Menge an möglichen Sätzen vergleicht. Das Ergebnis ist eine Schätzung, die gewandelt in eine vereinfachte Auszeichnungssprache an den GPSR-Knoten übergeben wird.

Dieses Konzept wurde aufgegriffen, um über die Android-Anwendung mit dem GPSR-Knoten zu kommunizieren. Die Anwendung nimmt für diesen Zweck die Position der Spracherkennung ein und übermittelt einen über die Benutzeroberfläche zusammengesetzte Aufgabenstellung an den GPSR-Knoten. Aus diesem Grund ist die Benutzerschnittstelle an den Aufbau eines natürlichen Satzes angelehnt (siehe Abbildung 18). Alle vom GPSR-Knoten angebotenen Aktivitäten können auf diese Weise ausgeführt werden. Gesprochene Rückfragen des Roboters zur Bestätigung eines Auftrags wurden unterbunden. Durch die Notwendigkeit einer vollständigen Definition der Aufgabe auf dem Endgerät wurde ein Dialog mit dem Roboter vermieden und der Roboter beginnt die unmittelbare Ausführung.

Auf diese Weise ist es möglich, den Roboter den Auftrag zu erteilen ein bekanntes Getränk zu bringen. Das Wissen über Orte, Objekte und möglicher Aktivitäten stellt der GPSR-Knoten zur Verfügung.

4.7 Bildschirmlayout

Für das Bildschirmlayout wurden zwei von Android angebotene elementare Konzepte gewählt: Fragments und Aktionsleisten. Das Layoutkonzept der Fragmente wurde in API-Level 11 eingeführt. Es handelt sich um eine Methode, die das Design von Benutzeroberflächen für Endgeräte mit unterschiedlichen Bildschirmgrößen vereinfacht. Fragmente erlauben den Entwurf von Layouts für unterschiedliche Bildschirmgrößen ohne große Änderungen einer bestehenden Hierarchie.
Ein Layout wird dabei nicht mehr direkt in einer Aktivität untergebracht, sondern in einem Fragment. Die Aktivität dient zur Darstellung von ein oder mehreren Fragmenten zur gleichen Zeit. Die Interaktion zwischen zwei Fragmenten in einer gemeinsamen Aktivität werden ebenso in der Stapelverarbeitung berücksichtigt, wie der Wechsel zwischen Aktivitäten. Durch diese Trennung ist das Erscheinungsbild einer Aktivität zur Laufzeit veränder- und steuerbar.

In der entwickelten Anwendung wird ein Fragment pro Autonomiestufe eingesetzt, welches eine Auswahl möglicher Aktionen zur Verfügung stellt. Wird die Anwendung auf einem Endgerät mit kleinem Bildschirm ausgeführt, so wird diese Liste bildschirmfüllend dargestellt. Die gewählte Aktion wird in einer nächsten Aktivität gestartet samt dem zugehörigen Fragment.
Auf einem großen Bildschirm werden die beiden Fragmente Seite an Seite dargestellt. Das beschriebene Verhalten ist in Abbildung 19 illustriert. Wählt der Anwender eine Aktion im ersten Fragment, so wird das benachbarte Fragment beendet und gegen ein Neues ausgewechselt. In beiden Fällen kommt die gleiche Programmlogik und das gleiche Layout zum Einsatz. Fragmente besitzen einen eigenen Lebenszyklus samt Rückfrage-Methoden und der Verarbeitung von Benutzereingaben.

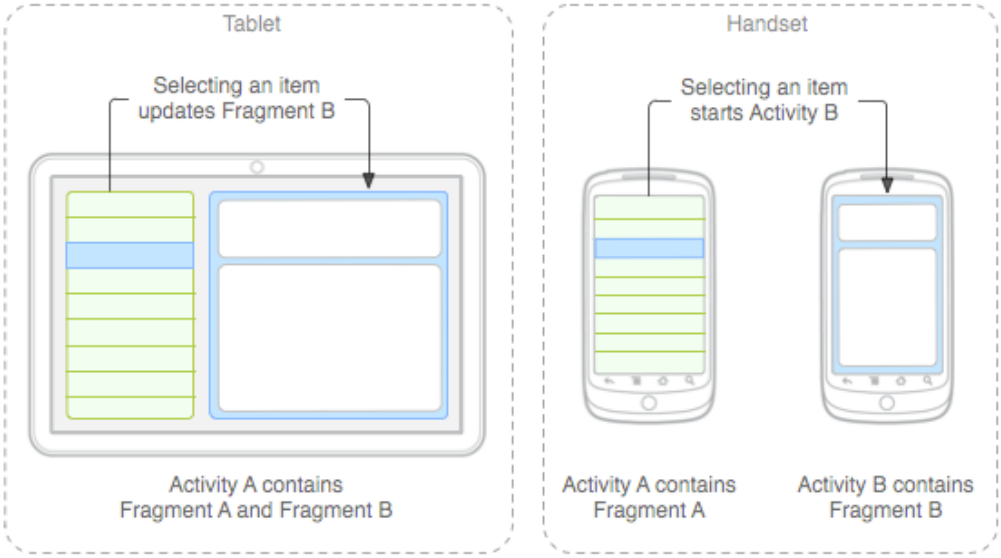


Abbildung 19: Darstellung auf unterschiedlichen Bildschirmgrößen [Proc]

Zur Auswahl des Autonomiegrades kommt eine sogenannte Aktionsleiste zum Einsatz. Sie ist am oberen Bildschirmrand angeordnet und vergleichbar mit einem Tabulator-Menü. Darüber hinaus sind in dieser Leiste programmweite Funktionen untergebracht, die über eine Berührung aktivierbar sind.

Der Start- und Stopp-Knopf dient zu einem temporären Pausieren des Roboters. Er ist notwendig, um die Aktivität des Roboters kurzfristig unterbrechen zu können. Beeinflusst wird von diesem Knopf nur die Bewegung der Basis durch die Deaktivierung des lokalen Planers und dem Senden eines Stopp-Befehls.
Um alle Gelenke des Roboters zu stoppen sollte die Funktionalität des bestehenden Notausschalters des Roboters genutzt werden. Ein roter Knopf im Nacken des Roboters. Er sorgt für die Fixierung aller Gelenke und das Ignorieren neuer Steueranweisungen, so lange der Schalter gedrückt ist.

Diese Funktionalität ist auf den eingesetzten Mikrocontrollern implementiert und bisher über keine Softwareschnittstelle verfügbar. Als langfristige Lösung sollte diese Funktion über die `robot_control` dem System zur Verfügung gestellt werden.

Ein weiterer Knopf dient zum Maximieren des Hauptfragments auf einem Endgerät mit großem Bildschirm. Es hat sich schnell gezeigt, dass das übergreifen der Auswahlliste im linken Fragment bei der Steuerung eines virtuellen Joysticks unangenehm ist. Insbesondere wenn dabei das Tablet in den Händen gehalten und nicht auf eine Unterlage gelegt wird. Dieser Knopf erlaubt das Minimieren der Auswahlliste und das gleichzeitige Maximieren des Fragments, in welchem die längerfristige Bedienung statt findet.

Den beiden Menüs zur Navigation durch die Funktionalität der Anwendung wurde eine Art Gedächtnis verliehen. Die Anwendung merkt sich dabei über einen längeren Zeitraum die zuletzt genutzten Eintrag und erlaubt so einen schnellen Wechsel zwischen Aktionen auf unterschiedlichen Autonomiestufen.

4.8 Bedienelemente

Als spezielle Bedienelemente zur Steuerung kommen sowohl virtuelle Joysticks als auch Gesten zum Einsatz.

Virtuelle Joysticks

Abbildung 20: Layout des virtuellen Joysticks [Koha]

Der eingesetzte virtuelle Joystick (siehe Abbildung 20) stammt aus der Bibliothek `android_honeycomb_mr2` und ist Bestandteil des *rosjava*-Stacks. Er wurde erweitert und modifiziert, um die notwendige Funktionalität zur Verfügung zu stellen. Teile dieser Änderungen sind zurück in das rosjava-Projekt geflossen und wurden in den Entwicklungszweig aufgenommen. Eine Erweiterung ist der holonome Modus, welcher für die Steuerung des omnidirektionalen Antriebs notwendig ist.
Darüber hinaus wurde der Joystick modifiziert, um die Steuerung der Blickrichtung des Roboters zu ermöglichen. Dazu ist der Aufruf eines Dienstes notwendig, anstatt der Veröffentlichung von Nachrichten an ein bestimmtes Thema.
Es hat sich herausgestellt, dass die Menge der vom Betriebssystem generierten Eingabeereignisse so hoch ist, dass sie nicht alle in Form von ROS-Nachrichten verschickt werden können. Ansonsten droht die Performanz einzubrechen und die Warteschlange für den Nachrichtenversand läuft über. Stattdessen kommen Zeitgeber (Timer) zum Einsatz, die für eine regelmäßige Veröffentlichung der aktuellen Auslenkung sorgen. Wird der Finger vom Joystick abgesetzt, so wird eine letzte Stopp-Nachricht veröffentlicht und der Nachrichtenversand pausiert. Dies entspricht den ROS-Konventionen für das Verschicken von Nachrichten, so dass weder Brandbreite verschwendet, noch die Funktion anderer Knoten beeinträchtigt wird.

Ein unterstützendes Merkmal des Joysticks ist das sogenannte Einschnappen. Es erleichtert das Drehen des Roboters auf der Stelle oder im holonomen Modus das seitliche Fahren. Der Einfluss einer versehentlich Bewegung entlang einer weiteren Achse wird verhindert. Berechnet wird dabei der Grad der Abweichung bei der Auslenkung des Joysticks von der x- und y-Achse. Beträgt die Abweichung von einer Achse weniger als 5 Grad, so wird die andere Komponente auf Null gesetzt.

Gesten

In weiteren sogenannten Ansichten, die als unsichtbare Ebene über das Kamerabild gelegt werden, wurden Gesten zur Interaktion mit dem Roboter implementiert. Android ist in der Lage unbegrenzt viele Finger auf dem berührungsempfindlichen Bildschirm zu erkennen. Die obere Grenze legt der Gerätehersteller fest. Die Spannweite erstreckt sich von der Erkennung von zwei Fingern bis hin zu Zehn. Einer, in einer Aktivität eingebetteten Ansicht, werden diese Informationen als Bewegungsereignis zur Verfügung gestellt. Das Bewegungsereignis, welches die Bewegung eines Fingers anzeigt stellt unter anderem den Aufsatzpunkt und die aktuelle Position des Fingers zur Verfügung. Mit Hilfe dieser Informationen wird die Auslenkung des Fingers bestimmt und als Steueranweisung an den Roboter übertragen.

5 Implementierung

Im folgenden Kapitel wird ein interessanter Aspekt der Implementierung herausgegriffen. Beschrieben wird die Methode der Anreicherung des übertragenen Kamerabildes mit zusätzlichen Informationen.

5.1 Live-Bild

Das vom Service-Roboter Cosero zur Verfügung gestellte Kamerabild stammt aus einer Microsoft Kinect. Eine RGB-D-Kamera, die ein Farbbild und ein registriertes Tiefenbild in Form einer Punktwolke mit Hilfe des Knotens `openni_camera` veröffentlicht. Um eine Übertragung des Kamerabildes über eine WLAN-Verbindung zu gewährleisten wurde die notwendige Bandbreite untersucht.

Das Nachrichtenformat „Image“ ist das Standardformat zur Übertragung von unkomprimierten Einzelbildern zwischen ROS-Knoten. Eine komprimierte Variante wird durch die Nachrichten „CompressedImage“ definiert. Dieser wird zur Übertragung von Bildern im JPEG- oder PNG-Format verwendet. Die Wandlung der Rohbilder in andere Bild- oder Videoformate erfolgt mit Hilfe des Knotens `image_transport`. Er beherrscht sowohl die Kompressionsverfahren JPEG und PNG, als auch das Übertragen von Bilderserien als Theora-Stream, indem einzelne Pakete in Nachrichten verpackt werden. Der Knoten ist plugin-basiert und erlaubt die flexible Erweiterung um neue Formate. Die Auflösung und Wiederholrate wird von der Bildquelle bestimmt und bleibt unverändert. Das Verfahren zur Kompression und die Stärke ist zur Laufzeit wählbar.

Durch Zuhilfenahme eines ROS-eignen Standardwerkzeugs (`rostopic bw /topic`) wurde die Bandbreite der unterschiedlichen Verfahren gemessen. Eine Serie unkomprimierter BGR8 kodierter Bilder bei einer Auflösung von 640×480 Bildpunkten und einer Wiederholrate von 30 Herz benötigt eine Bandbreite von 28 MB/s:

$$\underbrace{(640 \cdot 480)}_{\text{Auflösung}} \cdot \underbrace{(3 \cdot 8)}_{\text{BGR8}} \cdot \underbrace{30}_{\text{Herz}} \text{ bits/s.}$$

Auf einer 100BASE-TX Netzwerkleitung benötigt die Übertragung dieser Datenmenge 2.2 Sekunden. Unter gleichen Testbedingungen benötigt eine JPEG-komprimierte Bilderserie bei einer sogenannten Qualität von 80 Prozent die Bandbreite von 1.34 MB/s. Die Qualität gibt Auskunft über die Stärke der Quantisierung des Bildes.

Der Theora-Videostream bei einer Rate von 800.000 Bit liefert die geringste Bandbreite zwischen 18 KB/s und 110 KB/s. Der große Vorteil der Video-Codec Theora ist leicht zu erkennen. Es werden nicht nur Einzelbilder komprimiert, sondern ganze Serien von Bildern werden in die Kodierung einbezogen. Existieren nur leichte Veränderungen zwischen Einzelbildern, so kann an Bandbreite gespart

werden. Entsprechend höher ist die benötigte Bandbreite, wenn das Kamerabild starken Veränderungen ausgesetzt ist.

Vergleicht man die drei von ROS angebotenen Formate mit den unterstützenden Formaten eines Android-Geräts, so scheidet der Einsatz von Theora leider aus. Auch über eine zukünftige Implementierung von Theora schweigt sich Google aus. Die Rechte zum Einsatz dieser Codec sind grundsätzlich vorhanden. Ein Lichtblick ist der Plan der Implementierung von VP8/WebM in einer zukünftigen ROS-Version. Diese Codec wird ab Android 2.3.3 unterstützt und würde eine effiziente Übertragung von hochauflösenden Videostreams gewährleisten.

Als einzige für den Moment praktikable Lösung bleibt deshalb der Einsatz von komprimierten Einzelbildern. Die Verarbeitung komprimierter Bilder auf der Android-Plattform ist vergleichsweise einfach. Es wurde ein von einer *ImageView* abgeleiteter ROS-Knoten implementiert. Eine *ImageView* ist eine Ansicht zur Darstellung beliebiger Bitmaps. Die implementierte Funktionalität des ROS-Knotens sorgt für das Abonnieren von Nachrichten und dem Auslösen eines Ereignisses beim Empfang einer neuen Nachricht. Das komprimierte Bild wird dabei aus der Nachricht extrahiert, zu einem Bitmap gewandelt und an die *ImageView* übergeben. Der Aufruf der Methode *postInvalidate* sorgt für ein erneutes Zeichnen der Ansicht nach jedem Empfang und Setzen eines Bildes. Misst man die Häufigkeit des Zeichnens pro Sekunde lässt sich feststellen, dass eine Anzahl von 25 Bildern keine Belastung für ein Android-Tablet darstellt. Soll die benötigte Bandbreite weiter reduziert werden, so bietet sich das Verwerfen von Bildern an. Diese Aufgabe übernimmt der ROS-Knoten `throttle`.

5.2 Wahrnehmung dreidimensionaler Objekte

Das registrierte Tiefenbild der Kamera, in ROS verfügbar als Punktwolke, benötigt eine Bandbreite von 300 MB/s. Um mit diesen Informationen arbeiten zu können muss eine Vorverarbeitung stattfinden. Weder die Übertragung dieser Masse an Daten, noch deren Verarbeitung auf einem Android-Gerät ist praktikabel.

Eingesetzt wird die Tiefeninformation, um greifbare Gegenstände auf einer Tischplatte zu detektieren. Die dazu eingesetzte Segmentierung wurde bereits in Kapitel 4 erläutert. Das Ergebnis aus Segmentierung und Objekt-Tracker sind Nachrichten vom Typ *ObjectTrackingData*, welche unter dem Thema „/multi_object_tracker/object_tracks“ veröffentlicht werden. Der folgende Ausschnitt beschreibt den Nachrichtentyp im Detail.

```
Header header
geometry_msgs/Point32[] positions
geometry_msgs/Point32[] sizes
Matrix33[] principalComponents
float32[] confidences
```

```
float32[] thetas
string[] sources
uint32[] trackIDs
uint32 requestID
```

Aus dem Kopf der Nachricht ist zur Laufzeit das Koordinatensystem bestimmbar, in welchem sich die Daten befinden. Enthalten sind darüber hinaus die Positionen der gefunden Objekte im dreidimensionalen Raum, ihre Ausdehnung und das Ergebnis einer Hauptkomponentenanalyse, aus welcher sich die Orientierung des Objektes berechnen lässt. Die Konfidenz gibt Auskunft darüber, wie stark der Schätzung des Kalman-Filters vertraut werden darf. Das Array *trackIDs* enthält eine Identifikationsnummer pro Objekt und dient später zur eindeutigen Zuordnung.

Ziel ist die extrahierten Bounding Volumen in das Kamerabild auf dem Endgerät zu projizieren. Da sich herausgestellt hat, dass das Abonnieren von Transformationen viel Bandbreite beansprucht, findet eine weitere Vorverarbeitung auf dem Roboter statt. Ein für diese Aufgabe entwickelter Knoten namens `markerarray_transform` dient zum Empfang der beschrieben Nachricht. Die Objekte samt Orientierung und Ausdehnung werden dabei aus dem Ursprungskoordinatensystem *base_link* in das Kamerakoordinatensystem *openni_rgb_optical_frame* transformiert. Außerdem findet eine Selektion statt, so dass nur Objekte verarbeitet werden, deren Konfidenz besser als siebzig Prozent sind. Dies reduziert das kurzzeitige Einzeichnen von Artefakten oder ein temporäres Zerteilen von Objekten in zwei Segmente durch eine fehlerhafte Gruppierung. Die transformierten Daten erhalten ein neues Format. Sie werden als eine Liste visueller Marker veröffentlicht. Es handelt sich um ein Standardformat für die einfache Visualisierung von Daten in ROS. Über den Typ wird das Erscheinungsbild des Markers bestimmt. Die Identifikationsnummer eines Objektes wird auch zur Unterscheidung der Marker verwendet. Dies erlaubt eine einfache Darstellung in der ROS-eigenen Visualisierung *RViz* und hilft bei einer Fehlerdiagnose.

Diese Marker aus dem gleichen Koordinatensystem wie das Kamerabild werden an die Android-Anwendung übertragen. Auf der Anwendung müssen diese Nachrichten empfangen und dargestellt werden. Implementiert wurde für diesen Zweck eine Ansicht zur dreidimensionalen Darstellung von visuellen Markern. Die Klasse leitet sich von einer *GLSurfaceView* ab und implementiert eine *NodeMain*, das Java-Äquivalent eines ROS-Knotens, für den Nachrichtenempfang. Eine OpenGL Rendering Pipeline sorgt für die perspektivische Projektion der Marker.

5.3 Perspektivische Projektion

Bei der Aufnahme eines Kamerabildes wird eine dreidimensionales Umgebung in ein zweidimensionales Bildschirmkoordinatensystem transformiert. Jedoch entspricht das zweidimensionale Bild nicht einer perfekten Abbildung, es ist eine Transformation notwendig, um ein ideales Kamerabild zu erhalten.

In der Computergrafik existieren praktisch keine dreidimensionalen Bilder, sondern nur Zweidimensionale mit einer Illusion von Tiefeninformation. Um wirklich ein dreidimensionales Bild zu sehen, muss ein Objekt mit beiden Augen wahrgenommen werden. Das Augenpaar sieht ein jeweils zweidimensionales Bild aus unterschiedlichen Blickwinkeln. Das Gehirn fügt diese unterschiedlichen Bilder zu einem dreidimensionalen Bild zusammen.
Sehen beide Augen ein und dasselbe Bild, so muss die Abbildung trotzdem nicht flach erscheinen. Dies ist möglich, da viele dreidimensionale Effekte auch in einer zweidimensionalen Welt nachgebildet werden können. Der offensichtlichste Effekt ist, dass Objekte die weiter entfernt sind, kleiner erscheinen als nähere Objekte. In Kombination mit einem Blickwinkel wird diese Illusion der räumlichen Darstellung als Perspektive bezeichnet.

Die Projektion einer optischen Kamera wird mit dem vereinfachten Modell einer Lochkamera beschrieben. Dieses Modell ist in Abbildung 21 dargestellt.
Das Modell nimmt an, dass die Kamera statt einer Linse ein kleines kreisförmiges Loch (Lochblende) besitzt. Die Lochmitte ist das Projektionszentrum und ein Abbild der Realität wird auf die der Öffnung gegenüberliegende Bildebene projiziert. Die Größe der Öffnung, der Abstand zwischen Loch und Bildebene, sowie Art und Größe und Abstand zum aufgezeichneten Objekt bestimmen das entstehende Bild.

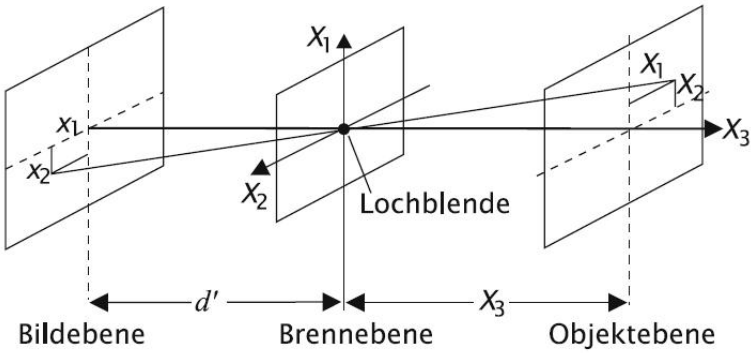


Abbildung 21: Lochkamera-Modell [Jäh05]

Alle Punkte $(X_1, X_2, X_3)^T$ des dreidimensionalen Weltkoordinatensystems lassen sich durch den aus dem Projektionszentrum einfallenden Lichtstrahl in der Bildebene $(x_1, x_2, -d')^T$ darstellen. Mathematisch besteht hierbei die folgende geometrische Beziehung:

$$x_1 = \frac{-d' \cdot X_1}{X_3}$$
$$x_2 = \frac{-d' \cdot X_2}{X_3}.$$

Die Variable d' wird auch mit f bezeichnet und ist die Brennweite einer Kamera, welche einen definierten Abstand zur Bildebene besitzt. Auch die Bildebene einer Kamera ist durch ihre Pixelanzahl und ihre Abmessungen definiert [Sch07].

Extrinisische Parameter Die extrinsischen Parameter definieren den Zusammenhang zwischen dem Kamerakoordinatensystem und dem Weltkoordiantensystem. Sie beschreiben die Position und Richtung der Kamera im Weltkoordiantensystem. Die Transformation M vom Weltkoordiantensystem besteht aus einer Rotation R und einer Translation t.

$$M = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}, \qquad t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Intrinsische Parameter Die intrinsischen Parameter definieren die Abbildung zwischen dem 3D-Kamerakoordinatensystem und dem 2D-Bildkoordinatensystem. Die inneren Parameter sind die Brennweite f, Pixelkoordinaten der Bildmitte $(0_x, 0_y)$, sowie die Pixelskalierung in horizontaler und vertikaler Richtung, s_x und s_y. Um die Abweichung des Bildkoordinatensystem von der Orthogonalität zu berücksichtigen wird ein zusätzlicher Parameter s (skew) eingeführt. Für normale Kameras und Abbildungen wird er aber meistens auf Null gesetzt.

$$K = \begin{pmatrix} f \cdot s_x & s & 0_x \\ 0 & f \cdot s_y & 0_y \\ 0 & 0 & 1 \end{pmatrix}$$

Die Matrix K heisst Kamera-Kalibrierungsmatrix und setzt voraus, dass sich die Kamera im Zentrum des euklidischen Koordinatensystems befindet.

Verzerrung Im Modell der Lochkamera erhält man eine „ideale Abbildung“, jedoch besitzen Kameras in der Praxis ein System aus Linsen. Dieses Herzstück jeder Kamera wird Objektiv genannt und erfüllt die Aufgabe einer Sammellinse. Die Kombination mehrerer Linsen dient zur Vermeidung von Farbfehlern und Verzerrungen. Dies ist der Grund für eine Verzerrung bei der Abbildung eines Punktes aus der Welt auf einen Bildpunkt.
Es gibt zwei wichtige Arten von Verzerrungen: radiale und tangentiale Verzerrung.

Radiale Verzerrung Die radiale Verzerrung skaliert den Abstand des Bildpunktes zum Projektionszentrum.

Tangentiale Verzerrung Die tangentiale Verzerrung (tangential zum Vektor aus dem Bildzentrum) resultiert aus der Dezentralisierung der Linsen der Objektive. Im Vergleich zur radialen Verzerrung ist der Einfluss der tangentialen Verzerrung gering.

Da die Verzerrung bei der eingesetzt RGB-D-Kamera äußerst gering ist, wird sie im folgenden vernachlässigt. Eine händische Bestimmung der Kamerakalibrierung ist nicht zwingend erforderlich. Sie ist möglich durch den Einsatz von OpenCV unter Zuhilfenahme eines Schachbrettmusters.

Die verwendete Kamera stellt die Informationen über in- und extrinisische Parameter über ihren Treiber zur Verfügung. Um auf dem Android-Handheld unabhängig von einer Kamera zu sein, bietet es sich an die Kalibrierungsmatrix über das Thema „camera_info“ zu beziehen. Die in OpenGL eingesetzte Projektionsmatrix zur Transformation von Weltkoordinaten in Bildkoordinaten wird mit Hilfe dieser Kamerakalibrierung erstellt.

Projektion von Weltkoordinalten in Bildkoordinaten Die Projektion eines Punktes m in Weltkoordinaten zu einem Punkt m' in Bildkoordinaten entspricht der folgenden Gleichung:

$$m' = K \cdot M \cdot m.$$

OpenGL-Projektionsmatrix Diese Projektion lässt sich wie folgt als OpenGL Projektionsmatrix darstellen:

$$\begin{pmatrix} 2 \cdot \frac{K[0,0]}{\text{width}} & -2 \cdot \frac{K[0,1]}{\text{width}} & \frac{\text{width} - 2K[0,2] + 2 \cdot x_0}{\text{width}} & 0 \\ 0 & -2 \cdot \frac{K[1,1]}{\text{height}} & \frac{\text{height} - 2 \cdot K[1,2] + 2 \cdot y_0}{\text{height}} & 0 \\ 0 & 0 & \frac{-\text{zfar} - \text{znear}}{\text{zfar} - \text{znear}} & -2 \cdot \frac{\text{zfar} \cdot \text{znear}}{\text{zfar} - \text{znear}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Dabei ist $K[n, m]$ ein Eintrag aus der Kalibrierungsmatrix K. Die Höhe (height) und Breite (width) beschreibt die Größe des Bildschirms, in welchen die Weltkoordinaten projiziert werden. (x_0, y_0) beschreibt den Kameraursprung. Die Parameter *zfar* und *znear* sind OpenGL-spezifische Parameter, um ein Sichtvolumen einer Szene zu definieren.

Diese OpenGL-Projektionsmatrix bestückt mit der intrinsischen Kamerakalibrierung erlaubt die Projektion dreidimensionaler Objekte in das Kamerabild. Implementiert wurde ein transparenter Quader, dessen Ausdehnung, Farbe und Position in Weltkoordinaten frei wählbar ist. Grundsätzlich kann ein beliebiges dreidimensionales Modell eingesetzt werden. Die OpenGL-ES-Implementation erlaubt auch das Laden von Objektbeschreibungen, so genannter Meshs. Empfängt

der Knoten zum Zeichnen der Marker eine Nachricht, so werden die in dieser Nachricht beschriebenen Bounding Volumen an den Renderer übergeben und in die Szene projiziert. Ein Beispiel für eine fertige Szene ist in Abbildung 22 dargestellt. Vereinzelt treten Abweichung zwischen dem realen Objekt und dem ins Bild projizierten Bounding Volumen auf. Auch ein sprunghaftes Verhalten der Orientierung ist zu beobachten. Beides sind Probleme auf Grund ungenauer Tiefeninformationen der RGB-D-Kamera und einer resultierenden falschen Schätzung der Segmentierung. Diese Artefakte sind ebenfalls in der ROS-eigenen Visualisierung zu finden und stammen nicht von einer fehlerhaften Kamerakalibrierung oder einer unzureichenden Genauigkeit der Projektion.

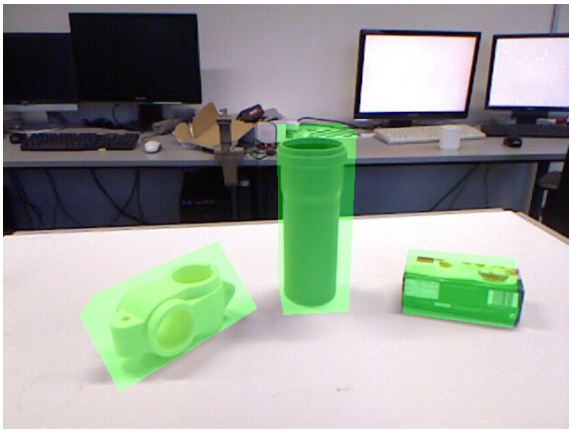


Abbildung 22: Projektion von Bounding Volumen

5.4 Selektion eines Objektes

OpenGL ES (OpenGL for Embedded Systems) ist ein Substrat einer OpenGL Implementation. OpenGL ist ein mit der Zeit gewachsener Standard, welcher so überladen ist, dass die Funktionsvielfalt für den Einsatz auf mobilen Geräten auf ein Minimum reduziert werden musste. Getrennt wurde sich in diesem Zusammenhang auch von einem speziellen Modus zur einfachen Selektion von Objekten in der dreidimensionalen Szene.

Android bietet ebenfalls keine direkten Hilfsmittel. Die eingesetzte Ansicht erlaubt zwar das Abfangen von Eingabeereignissen, jedoch werden dabei keine Informationen über die gezeichnete Szene zur Verfügung gestellt. Es gibt zwei Ansätze dieses Problem zu lösen.

Strahlverfolgung Bei der Strahlverfolgung (Raytracing) wird die Richtung der Projektion umgekehrt. Zum Einsatz kommt die Funktion gluUnProject. Eine von sechs auf der Android-Plattform verfügbarer Methoden des OpenGL Utility Toolkits. Sie erlaubt die Transformation einer zweidimensionalen Bildkoordinate in einen Lichtstrahl durch in die dreidimensionale Welt geworfen wird.

Anschließend wird geometrisch die Kollision dieses Lichtstrahls mit durchstoßenen Objekten geprüft.

Farbwahl Die Farbwahl wurde ebenfalls implementiert und führte zu einem akkuraten Ergebnis. Bei dieser Methode werden Objekte unterschiedlich eingefärbt (siehe Abbildung 23). Die eingesetzte Farbkodierung (RGBA8888) erlaubt eine Vielfalt von 255^4 Farben. Dies genügt für eine ausreichende Differenzierung unterschiedlicher Objekte. Die Auswahl eines Objekts erfolgt über die Bildschirmkoordinate und einen Vergleich mit dem Bildspeicher (Framebuffer).

Für den Anwender ist es unangenehm mit Objekten in zufälligen Farbgebungen zu arbeiten. Anstatt einen zweiten Puffer einzusetzen und zur gesamten Laufzeit zu pflegen wird ein einzelnes spezielles Bild für die Farbwahl im normalen OpenGL-Puffer gezeichnet, wenn eine Berührung des Bildschirms stattgefunden hat. Zum zeichnen dieses Einzelbildes werden Farbe verfälschende Effekte deaktiviert. Objekte sind an Hand ihrer Identifikationsnummer eingefärbt und auf einen schwarzen Grund gezeichnet. Die Identifikationsnummer des ausgewählten Objektes kann im weiteren Verlauf eingesetzt werden, um einen Greifvorgang einzuleiten.

(a) (b) (c)

Abbildung 23: Auswahl unterschiedlicher Objekte

6 Evaluation

Benutzbarkeit bezeichnet nach ISO/TR 16982:2002 Standard, wie gut ein Produkt zu gebrauchen ist. Damit ist gemeint, wie gut ein bestimmter Benutzer unter einem bestimmten Kontext ein bestimmtes Ziel erreicht. Dieses Ziel soll dabei effizient, effektiv und zufriedenstellend erreicht werden. Gute Benutzbarkeit trägt in vielen Fällen dazu bei, Kosten zu senken. Es werden Fehler reduziert, Arbeitsvorgänge beschleunigt und der Bedarf an Support oder Schulungen gesenkt. Das Benutzbarkeit sowohl subjektiv als auch nicht messbar ist oder entwickelt werden kann, ist ein weit verbreiteter Mythos unter Entwicklern. Benutzbarkeit ist keine Mischung aus subjektiver Meinung, ästhetischen Gesichtspunkten und Menschenverstand.

Um die Benutzbarkeit der implementierten Anwendung zu untersuchen werde ich ihre Güte an Hand von fünf zentralen Faktoren bemessen. Diese Faktoren sind Erlernbarkeit, Effizenz, Fehlerrate, Rentention und subjektive Zufriedenheit des Anwenders. Diese Kriterien und ihre Auswertung orientieren sich an Seffah et al. [SDKP06].

Die Evaluation der in dieser Arbeit entwickelten Android-Anwendung erfolgte summativ, nach Abschluß der Programmierung. Um die Benutzbarkeit bewerten zu können, wird ein Benutzerprofil definiert. Bei der Entwicklung der Anwendung wurde dieses Profil ebenfalls berücksichtigt. Es ist zum Beispiel nicht zielführend in einer Arbeitsumgebung, in welcher Handschuhe getragen werden müssen einen kapazitiven Touchscreen einzusetzen. Ebenso wenig sollte in der Wüste während eines Sandsturmes die Kommunikation mit einem Roboter über Gesten erfolgen.
Auf Grund der Komplexität sind HRI-Systeme sehr schwer quantitativ zu evaluieren und zu vergleichen. Deshalb ist es umso wichtiger zur Evaluation Standards zu besitzen, um besondere Schlüsselfaktoren zu identifizieren.

Im ersten Schritt hat eine heuristische Evaluation stattgefunden. Clarkson und Arkin [CA07] haben Usability-Heuristiken für die Evaluation von HRI-Systemen entwickelt. Hierfür wurden aus einer Vielzahl von Quellen in und außerhalb des HRI-Anwendungsgebietes Heuristiken zusammengetragen und für die Robotik angepasst. Mit Hilfe dieser Methoden war es ihnen möglich mit nur $3 - 5$ Testpersonen $40 - 60\%$ bereits bekannter Probleme zu identifizieren. Vergleiche Abbildung 24.

Die erste Evaluation wurde deshalb mit einzelnen Personen aus der Robotik- und Softwareentwicklung durchgeführt. Dabei wird den Experten ein horizontaler Prototyp der Anwendung vorgelegt, welcher die Benutzerschnittstelle abbildet, jedoch keine weitere Funktionalität besitzt.
Auf diese Weise wurden zukünftige Benutzer frühzeitig eingebunden und die Spezifikationen und Benutzbarkeit der Anwendung validiert. Es sollten sowohl frühzeitig

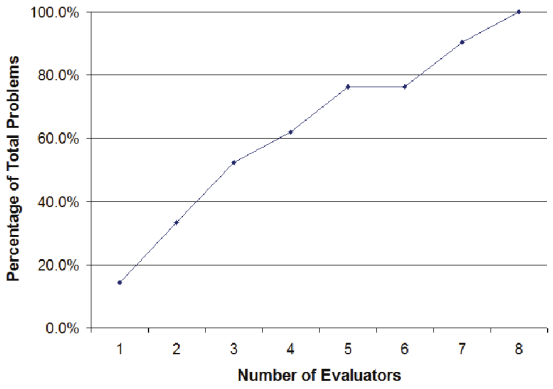


Abbildung 24: Prozentsatz gefundener bekannter Probleme bei steigender Zahl an Testpersonen [CA07]

Probleme aufgezeigt werden, als auch Korrektur- und Verbesserungsvorschläge eingeholt.

6.1 Benutzerprofil

Das folgende Benutzerprofil trägt die Bezeichnung „eingeschränkte Mobilität“. Die Benutzerrolle beschreibt eine in seiner Mobilität eingeschränkte Person. Der Benutzer ist dabei an ein Bett gebunden und besitzt keine Möglichkeit dieses zu verlassen. Ein persönlicher Service-Roboter, der über ein Android-Gerät bedient wird, soll dem Benutzer einen Teil seiner Selbstständigkeit zurückgeben. Dabei soll der persönliche Service-Roboter Aufgaben erledigen, die der Anwender auf Grund seiner eingeschränkten Mobilität nicht selbst ausüben kann. Das Hauptziel ist das Bringen von Gegenständen an das Bett des Anwenders.
Der Anwender ist ausschließlich in seiner Mobilität eingeschränkt. Die Häufigkeit der Nutzung ist nicht eingeschränkt oder weiter spezifiziert. Sie kann so lang erfolgen, wie der Roboter läuft oder es weitere Randbedingungen erlauben.

Das System muss in der Lage sein, sich in seiner Umgebung zurecht zu finden, zu ausgewählten Räumen zu navigieren, Hindernisse zu vermeiden, einen Gegenstand zu greifen und diesen zu übergeben.
Dabei müssen folgende Randbedingungen erfüllt werden: Der Roboter darf sich nicht außerhalb der Reichweite des WLAN-Netzes bewegen. Die Umgebung, in welcher der Roboter navigiert, muss eben sein und darf keine Stufen enthalten. Die zu greifenden Gegenstände müssen für den Roboter gut zugänglich sein. Sie dürfen sich nicht verdecken oder zu nah beieinander stehen.

Die Computererfahrung des Anwenders darf durchschnittlich sein.

6.2 Szenario

Mit den Testpersonen wird eine Blickübung, eine Fahrübung und eine Greifübung durchgeführt. Jede Übung wird zuerst auf der ersten Autonomiestufe bearbeitet, anschließend auf der zweiten Stufe. Die Umgebung, in der sich der Roboter bewegt wird den Testpersonen kurz vorgestellt. Sie haben die Möglichkeit sich einen Eindruck über das Umfeld zu verschaffen. Dies soll dazu beitragen indirekt das Benutzerprofil besser zu erfüllen. Im beschriebenen Profil wird der Roboter in einer bekannten Umgebung bewegt. Auf Grund einer unbekannten Arbeitsumgebung soll der Anwender keine zusätzliche Angst oder überdurchschnittliche Vorsicht entwickeln. Während der Übungen besitzt die Testperson keinen direkten Blick auf den Roboter und die umgebene Szene. Jegliche Wahrnehmung und die Pflege des Situationsbewusstseins erfolgt über die Benutzerschnittstelle. Die Benutzer werden dazu aufgefordert das Tablet im Sitzen zu bedienen. Darüberhinaus wurden die Testpersonen zum lauten Denken aufgefordert. Sie sollen dabei ihr Vorgehen zur Lösung der Aufgabe kommentieren. Da diese Methode nicht jeder Testperson leicht fällt, wurden sie insbesondere bei der Unterbrechung ihrer Tätigkeit, wie das Pausieren während der Steuerung, aufgefordert ihre Gründe zu benennen.

Blickübung Im Rahmen einer kleinen Übung, soll der Benutzer versuchen sich mit Hilfe des Roboters einen Eindruck von der Umgebung zu verschaffen in der sich der Roboter befindet. Die Testperson wird dazu aufgefordert in die entsprechende Ansicht zur Steuerung des Blicks zu wechseln und die unterschiedlichen Bedienelemente zu erproben. Versucht die Testperson nicht aus eigenem Antrieb den Einsatz von Gesten, so wird sie aufgefordert diese Art der Interaktion ebenfalls auszuprobieren.

Fahrübung Der Roboter befindet sich an einer festen Startposition im Raum und soll sich zu einem definierten Ziel in fünf Meter Entfernung bewegt werden. Im ersten Teil der Übung ist die zurückzulegende Strecke eine gerade Linie und kann ohne zusätzliche Manöver auf direktem Weg erreicht werden. Da das Experiment in einem Büroraum durchgeführt wird, ist der Weg von Tischen gesäumt, die als entfernte Hindernisse berücksichtigt werden müssen. Im zweiten Teil wird das Experiment wiederholt. Diesmal wird der direkte Weg durch ein Hindernis in Form eines Kartons versperrt. Da es sich um ein dynamisches Hindernis handelt, ist es nicht in der Karte des Roboters vermerkt. Die Testperson erhält erneut den Auftrag den Roboter zum Ziel zu steuern. Das Hindernis ist dabei so platziert, dass der direkte Blick auf das Ziel nicht verdeckt wird. In einem dritten Versuch muss die Aufgabe über die zweite Autonomiestufe der Anwendung gelöst werden. Der Anwender setzt dazu einen Zielpunkt im Kamerabild. Die vollständige Navigation zum Ziel und die Umfahrung des Hindernisses erfolgt vollständig autonom. Die Geschwindigkeit des Roboters ist für das gesamte Experiment auf 40 Zentimeter pro Sekunde limitiert.

Greifübung Bei der Greifübung steht der Roboter vor einem Tisch mit gesenktem Kopf und Blick auf die Tischplatte. Auf dem Tisch befinden sich

unterschiedliche für den Roboter einfach zugängliche Objekte. Die Testperson wird aufgefordert in der zweiten Autonomiestufe ein vorgegebenes Objekt auszuwählen. Die Vorgabe ist notwendig, um die unterschiedlichen Ergebnisse der Testpersonen besser vergleichen zu können. Das Objekt steht dabei möglichst ideal, so dass nur eine minimale Neuausrichtung der Basis stattfindet. Der Greifvorgang und das Anheben des Objektes erfolgt autonom.

Im zweiten Teil der Greifübung wird das Objekt an seinen Platz zurück gestellt. Die Erreichbarkeit des Objektes ohne eine Neuausrichtung der Basis durch die Testperson ist damit gewährleistet. Es ist notwendig, um die Komplexität des Experiments und die Anzahl der möglichen Fehler oder Probleme minimal zu halten. Durch die wiederholte Ausführung ist die Testperson darüber informiert, dass der Roboter nicht bewegt werden muss. Die Testperson wird aufgefordert den Greifvorgang über die erste Autonomiestufe manuell zu steuern. Die Ausgangsposition des Armes beschreibt dabei, wie auch im ersten Teil der Experiments, eine sogenannte bequeme Haltung. Dabei lässt der Roboter die Arme leicht angewinkelt hängen. Die Hände ergeben zusammen mit dem Unterarm eine gerade Linie. Dies hat zur Folge, dass die Testperson gezwungen ist auch die Orientierung der Hand zu steuern, um eine Greifbewegung parallel zur Tischplatte auszuführen. Wurde das vorgegebene Objekt angehoben, so gilt die Aufgabe als erfüllt.

6.3 Mensch-Roboter-Interaktion

Mit Hilfe der drei Experimente werden unterschiedliche Aspekte der Mensch-Roboter-Interaktion analysiert. Ziel ist die entwickelte Anwendung zur Interaktion mit dem Roboter auf ihre Benutzbarkeit untersuchen.

Eine quantitative Evaluation lässt kaum Spielraum für menschliche Subjektivität und legt einen engen Fokus auf einzelne klar definierte Thesen. Auch die hohe Komplexität des zu evaluierenden HRI-Systems und die notwendige hohe Anzahl an Testpersonen für ein signifikantes Ergebnis spricht gegen eine quantitative Evaluation. Stattdessen wird die entwickelte Anwendung mit Hilfe eines qualitativen Verfahrens untersucht. Diese Methode ist weniger strikt, fördert Vielfalt und erlaubt Subjektivität. Auf diese Weise sollen neue Ansätze entdeckt und wichtige Aspekte hervorgehoben werden.

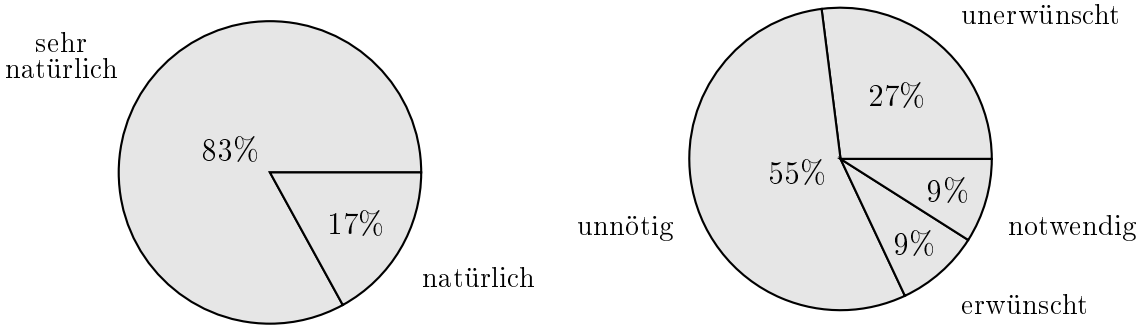
Blickübung Im Rahmen der Blickübung greift der Benutzer zur Steuerung des Roboterkopfes auf einen Joystick oder Gesten zurück. Beim Joystick handelt es sich um ein etabliertes Bedienelement zur Steuerung, während die Wischgeste ein Steuerelement ist, das zwar aus dem Umgang mit Computern, in Form eines Scrollvorgangs bekannt ist, jedoch hauptsächlich durch Handhelds und ihre berührungsempfindlichen Bildschirme Verbreitung gefunden hat. Es soll bewertet werden, ob der Einsatz von Gesten dem Anwender zusagt und eine natürliche Steuerung des Roboters erlaubt.

Fahrübung Über das Experiment der Fahrübung wird das Konzept der unterschiedlichen Autonomiestufen untersucht und bewertet. Hierbei werden sowohl die objektiv benötigte Interaktions- und Bearbeitungszeit zum Lösen der Aufgabe erhoben, als auch die subjektive Meinung des Nutzers in die Wertung einbezogen. Dies soll eine Auskunft geben über die Arbeitsbelastung des Anwenders, seine Zufriedenheit und die Erlernbarkeit der Anwendung.

Greifübung Mit Hilfe der Greifübung wird sowohl das Situationsbewusstsein der Testperson geprüft, als auch die unterschiedlichen Autonomiestufen quantifiziert. Diese Art der Interaktion ist weitaus komplexer und Bedarf einer größeren Aufmerksamkeit. Über die subjektive Arbeitsbelastung dürfen wiederum Rückschlüsse auf die Bedienbarkeit der Anwendung gezogen werden.

6.3.1 Gemeinsamkeiten

Der Dienstleistungsroboter wird ein fester Bestandteil in einem Haushalt sein und eine in seiner Mobilität eingeschränkte Person wird dem Roboter viel Geduld entgegen bringen. Es ist nicht notwendig, dass die Funktionalität des Roboters innerhalb von kürzester Zeit erlernt und überblickt wird.
Demnach sollte die Nutzbarkeit des Android-Handhelds möglichst intuitiv gestaltet sein. In den Tests stellte sich heraus, dass die meisten Testpersonen die Funktionsweise des Joysticks zur Veränderung der Blickrichtung sofort klar war (vergleiche Abbildung 25a) und sie kein zusätzliches visuelles Feedback zur Steuerung des Blickes benötigen (vergleiche Abbildung 25b).



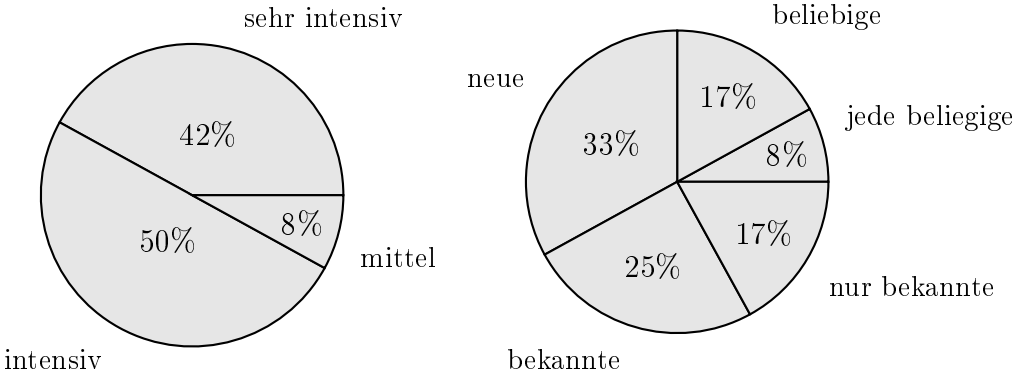
(a) Empfundene Natürlichkeit der Gestenmanipulation

(b) zusätzliches visuelles Feedback

Abbildung 25: Gestenmanipulation

Wie aus Abbildung 26 hervor geht, sprachen sich alle Benutzer dafür aus, dass Gesten intensiv eingesetzt werden sollen, jedoch waren nur 25 Prozent der Testpersonen bereit jede beliebige Geste zu erlernen.

Der Joystick konnte präzise eingesetzt werden. Die Nutzung der Gesten hingegen



(a) Nutzung (b) Gestenart

Abbildung 26: Einsatz von Gesten und Gestenarten

war intuitiver aber auch ungenauer. Auch beurteilten die Testpersonen, dass die virtuellen Joysticks die Sicht nicht sonderlich einschränken, aber auch keinen großen Mehrwert bieten.

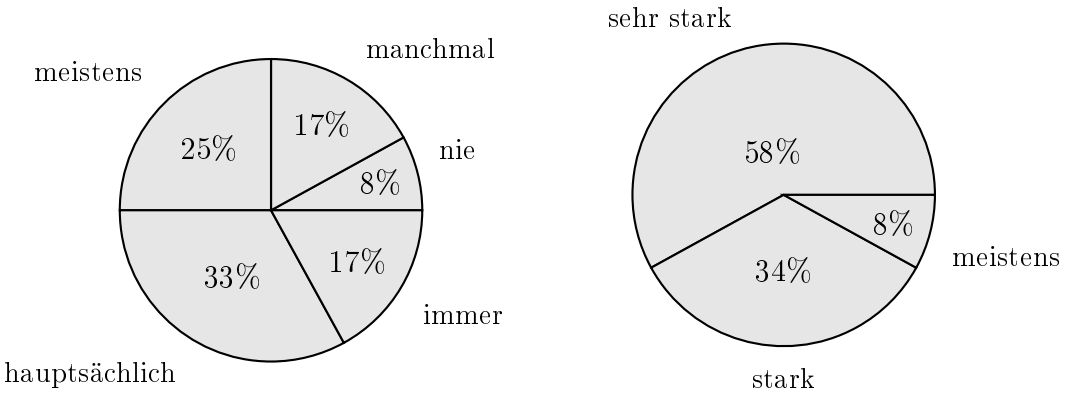
6.3.2 Vernachlässigungstoleranz

Die Vernachlässigungstoleranz ist eine Möglichkeit den Grad der Autonomie des Roboters zu messen. Sie erlaubt außerdem einen Rückschluss auf die objektive Arbeitsbelastung der Testperson und die Einschätzung eines gewonnen Mehrwerts durch den Roboter. Die Voraussetzung ist ein Vertrauen in den Roboter. Wird der Roboter während der Bearbeitung der Aufgabe nicht aus den Augen gelassen, so ist der Mehrwert der Autonomie zweifelhaft. Bei der Entwicklung einer Teleoperationsanwendung sollte sich also auf eine abgesicherte Teleoperation konzentriert werden.

Abgefragt wurde die subjektive Wahrnehmung des Anwenders, der dem Roboter ein Vertrauen entgegen bringen muss, um ihn eine Aufgabe selbstständig und unbeobachtet ausüben zu lassen. Ein objektives Maß ist die eingesparte Interaktionszeit. Dazu wird die Interaktionszeit mit der Bearbeitungszeit einer Aufgabe verglichen.

In der Befragung (vergleiche Abbildung 27a) gab die Hälfte der Testpersonen an, dass sie sich sehr stark darauf verlassen, dass der Roboter Gefahren verhindert. Auch waren die Testpersonen sich darüber einig, dass sie dem Roboter so weit vertrauen, dass sie ihn unbeobachtet agieren lassen (vergleiche Abbildung 27b).

Fahrübung Die Aufgabe des Anwenders ist das Abfahren eines gekennzeichneten Weges. An Hand der Abweichungen von diesem Weg und das subjektive Empfinden



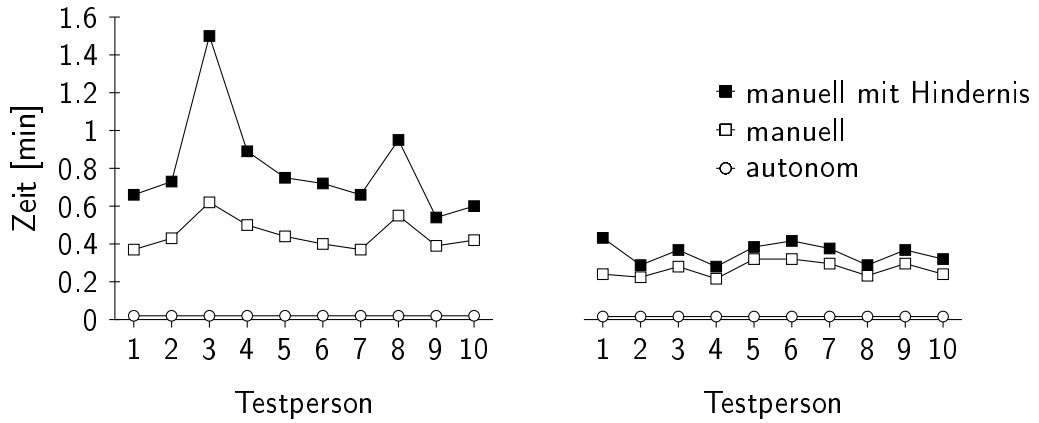
(a) Verlassen (b) Vertrauen

Abbildung 27: Wertschätzung des Roboters und seiner Fähigkeiten

des Anwenders wird eine Arbeitsbelastung gemessen. Um die Belastung des Anwender schrittweise zu erhöhen gibt es drei unterschiedliche schwere Testanordnungen.

- Navigation aus der Entfernung auf einem gekennzeichneten Weg ohne Hindernisse
- Navigation aus der Entfernung auf einem gekennzeichneten Weg mit einem Hindernis
- Navigation aus der Entfernung auf einem gekennzeichneten Weg mit einem Hindernis (sichere Teleoperation)

Als weitere Option wird die sichere Teleoperation eingesetzt und die Auswirkungen auf die Belastung des Anwenders untersucht.



(a) unerfahrene Nutzer (b) erfahrene Nutzer

Abbildung 28: Interaktionszeit während der Fahrübung

Navigation aus der Entfernung auf einem gekennzeichneten Weg ohne Hindernisse Im Durchschnitt brauchten die Nutzer 0.7 Minuten um von einem Punkt zum Anderen zu navigieren. Alle Testpersonen haben angegeben, dass sie den Roboter fast die komplette Aufgabenzeit bewachen mussten.

Navigation aus der Entfernung auf einem gekennzeichneten Weg mit einem Hindernis Durch das Hinzufügen eines Hindernisses zeigten sich deutliche Einbußen in der empfundenen Roboterkontrolle. Auch wurden die durchschnittliche Bearbeitungszeit 25 Prozent länger.
Deutlich wurden in diesem Test die Unterschiede zwischen Testpersonen mit Erfahrungen in Computerspielen, First-Person-Shootern oder Teleoperation (siehe Abbildung 28).

Navigation aus der Entfernung auf einem gekennzeichneten Weg mit einem Hindernis (sichere Teleoperation) Die Interaktion der Testpersonen mit dem Roboter beschränkte sich auf einen einzelnen Klick mit dem die Zielposition des Roboters im Kamerabild bestimmt wurde. Diese Aktion war unabhängig von den Vorkenntnissen sehr einfach auszuführen.

Auch gaben die Testpersonen an, dass sie sich die Entlastung durch den Roboter mittels autonom durchgeführter Teilaktionen wünschen (siehe Abbildung 29).

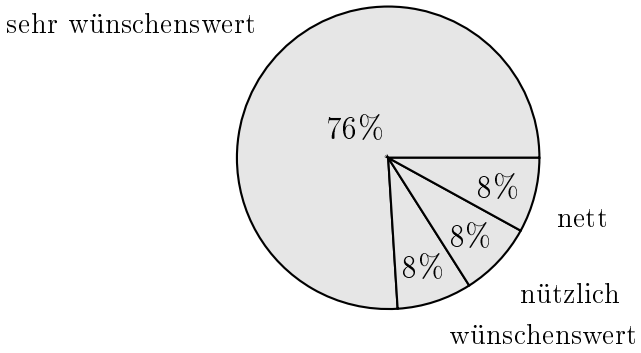


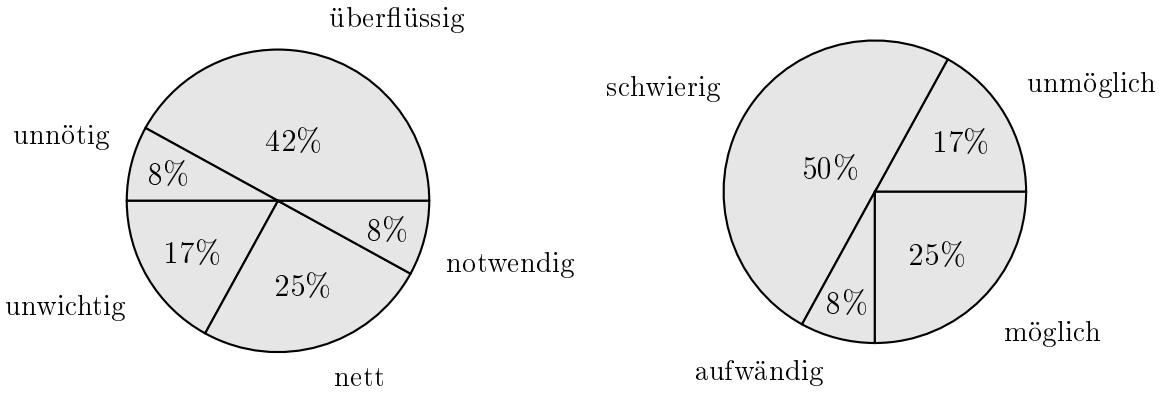
Abbildung 29: Nützlichkeit der autonomen Ausführung von Teilaktionen

6.3.3 Situationsbewusstsein

Um das Situationsbewusstsein zu testen wurde neben der Fahr- und Greifübung auch noch konkrete Fragen zum Empfinden der Situation und deren Ursache gestellt.

Fahrübung Es wurde deutlich, dass den Testpersonen, die zur Verfügung gestellten Informationen nicht ausreichten um ein umfassendes Situationsbewusstsein zu

erhalten. Zwar gaben die Testpersonen an, dass die Visualisierung eines Laserscans nicht nötig sei, um sich sicher zu fühlen (siehe Abbildung 30a). Dennoch reichte das Kamerabild alleine nicht aus um Kollisionen zu erkennen (siehe Abbildung 30b).



(a) Notwendigkeit von Laserscans

(b) Kollisionserkennung

Abbildung 30: Situationsbewusstsein Aufgrund des Kamerabilds

Greifübung Eine Benutzerschnittstelle beeinträchtigt das Situationsbewusstsein eines Anwenders, wenn er mit der Bedienung dieser überfordert ist. Evaluiert wird deshalb die Belastung des Anwenders bei der direkten Steuerung des Roboters über den virtuellen Joystick.
Grundsätzlich sei angemerkt, dass der Roboter Cosero als autonomer Haushaltsroboter entwickelt wurde und somit einige Mängel in der Benutzung mittels Teleoperation aufweist. Dies spiegelt sich unter anderem in dem für die Teleoperation nicht sehr geeigneten einfachen Kamerabild wieder. Die Testpersonen bemängelten ein nicht ausreichendes Situationsbewusstsein, da das Kamerabild keinerlei Tiefeninformation zur Verfügung stellt.

Während der Roboter nach dem Anklicken des Objektes nur eine halbe Minute braucht um den Gegenstand zu greifen, benötigten die Anwender ohne Erfahrung mit Ego-Shootern oder der Teleoperation von Robotern im Schnitt fünfmal länger, bei der manuellen Steuerung.

Obwohl 60 Prozent der Anwender Spaß an der manuellen Steuerung des Arms hatten, wurde die Aufgabe zwar nicht als kompliziert aber als langwierig empfunden (siehe Abbildung 32). Auch hier zeigte sich ein großer Unterschied zwischen erfahrenen und unerfahrenen Testpersonen (vergleiche 31).

Auch fühlten sich die Testpersonen während des teleoperierten Greifens sehr stark belastet und unsicher, während die Bedienung des halb-automatischen Greifens als leicht und sicher empfunden wurde (vergleiche Abbildung 33).

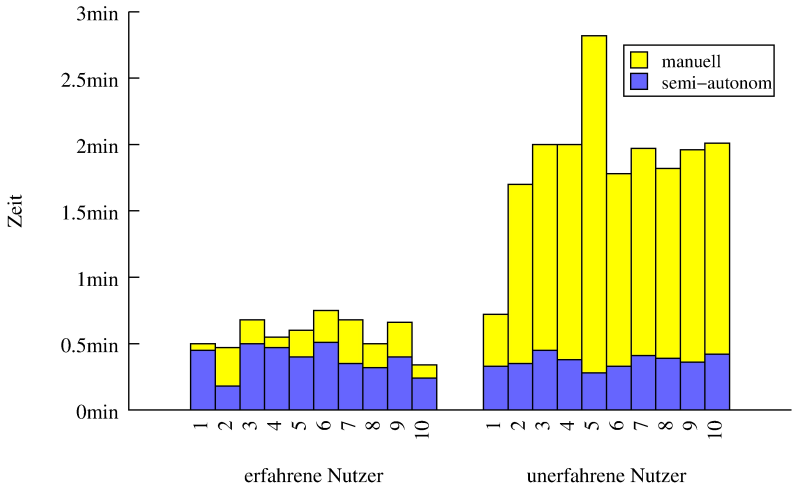
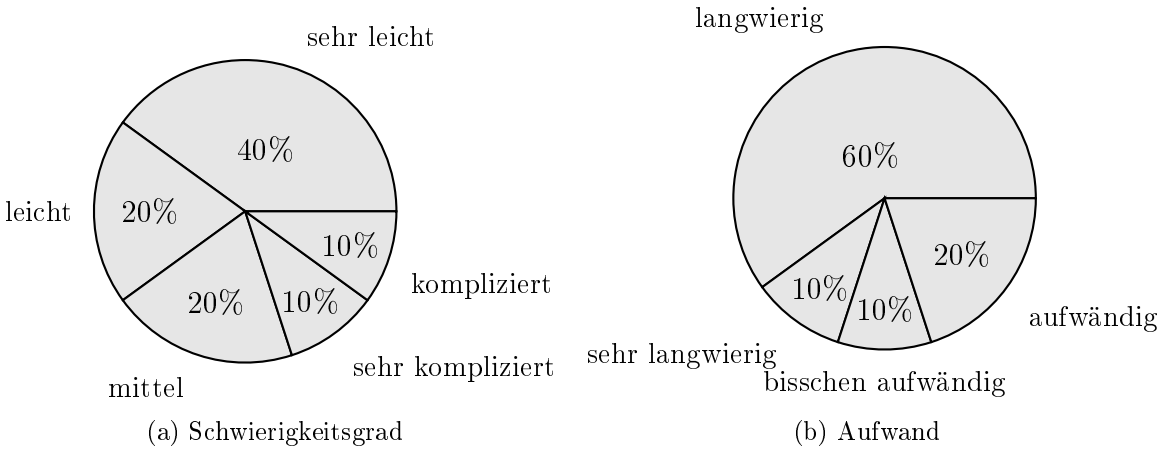


Abbildung 31: Bearbeitungszeiten für ausgewählte Testpersonen



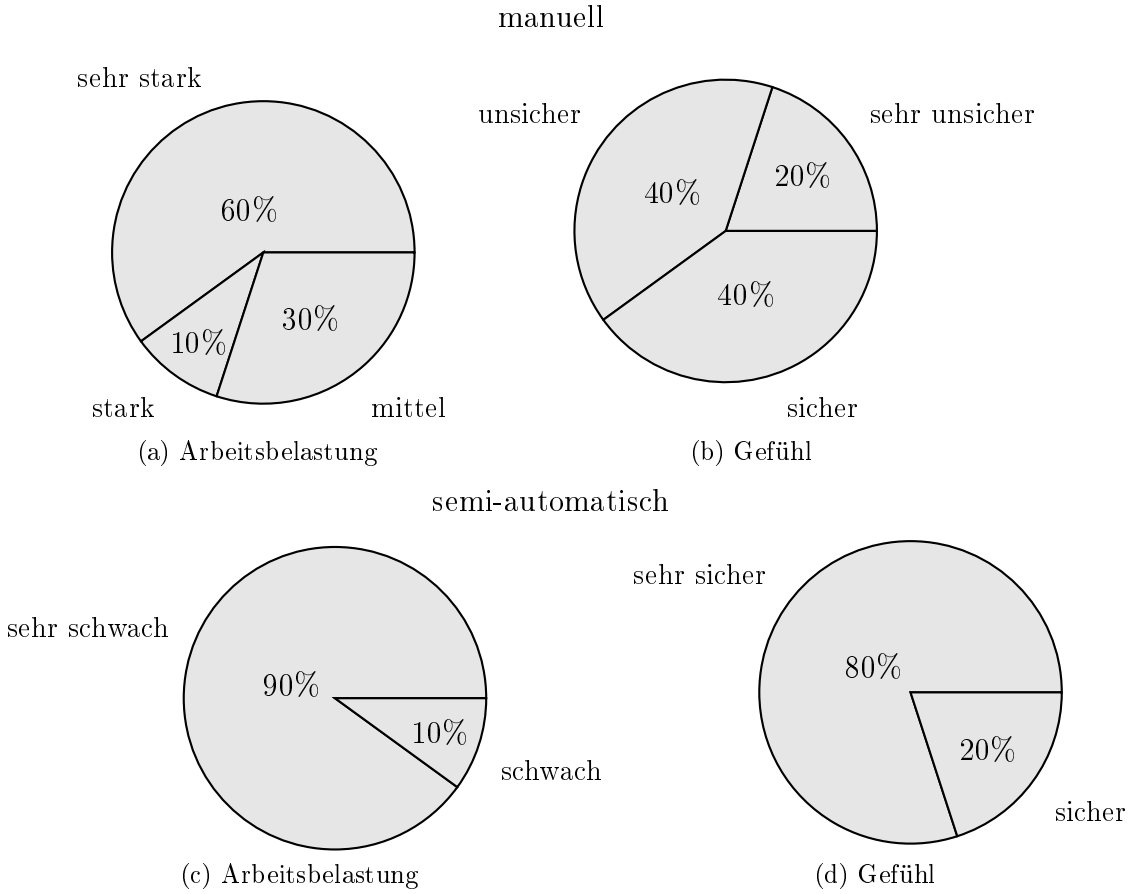
(a) Schwierigkeitsgrad

(b) Aufwand

Abbildung 32: Zufriedenheit mit der Bedienbarkeit des Greifarms

Es ist deutlich zu sehen, dass selbst erfahrene Nutzer von der Autonomie entlastet werden. Das Grundverständnis der Steuerung des Arms mit Hilfe von vier Joysticks ist deutlich komplexer als die einfache Steuerung der Fahrtrichtung. Die Autonomie maskiert die Komplexität der Bedienung von sechs Freiheitsgraden.
Der Unterschied zwischen der rein manuellen und der halb-autonomen Steuerung kommt bei längeren Bearbeitungszeiten deutlicher zum Vorschein als bei der kurzen Fahrübung.

Obwohl den Nutzern die Tiefeninformation fehlte, waren sie durchaus bereit, diese spielend zu erkunden. Dafür wäre nur ein weiteres Feedback nötig gewesen. So wäre es wünschenswert, wenn der Nutzer in Form eines visuellen oder haptischen Feedbacks mitgeteilt bekommt, wenn die Roboterhand den Tisch berührt. Eine weitere vorgeschlagene Alternative war das Einzeichnen



(a) Arbeitsbelastung (b) Gefühl

(c) Arbeitsbelastung (d) Gefühl

Abbildung 33: Steuerung des Roboterarms

von Tiefeninformation in das Kamerabild in Form von Gitterlinien oder Graustufen.

Im Gegensatz zur Tiefeninformation waren sich die Testpersonen einig, dass das Kamerabild zur Objektidentifikation perfekt geeignet ist (siehe Abbildung 34).

6.3.4 Zufriedenheit

Die Befragung der Testpersonen hat ergeben, dass die Anwender durchaus zufrieden mit der Bedienbarkeit des Roboters mit Hilfe des Android-Handhelds sind. Die Gestaltung der Anwendung sei einheitlich und die Anordnung der Steuerelemente schlüssig (siehe Abbildung 35).

Auch seien die verwendeten Steuerelemente verständlich und führten stets zum erwarteten Ergebnis (siehe Abbildung 36).

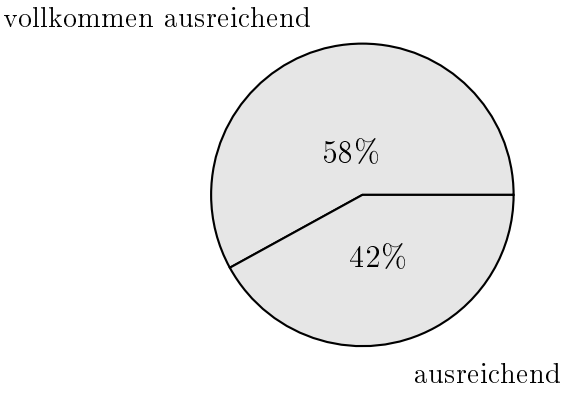
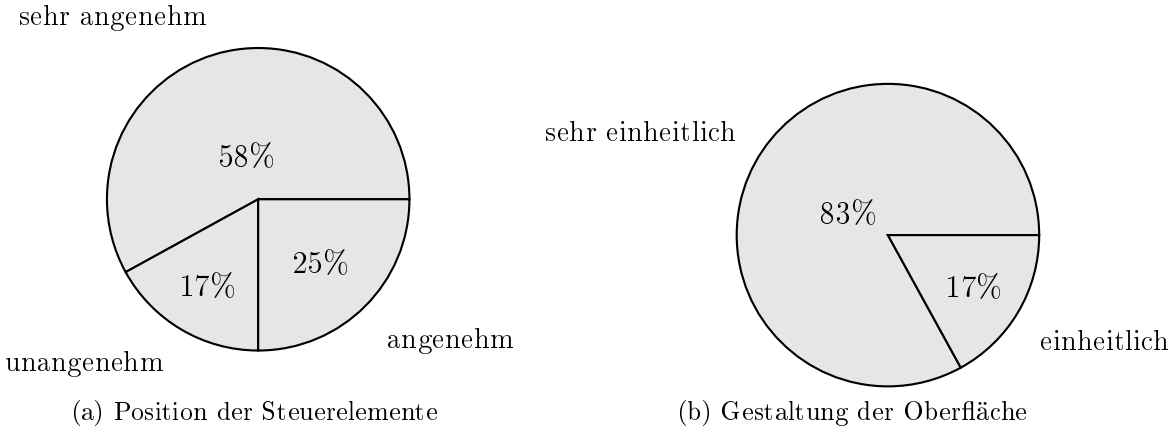


Abbildung 34: Qualität der Objekterkennung mittels Kamerabild



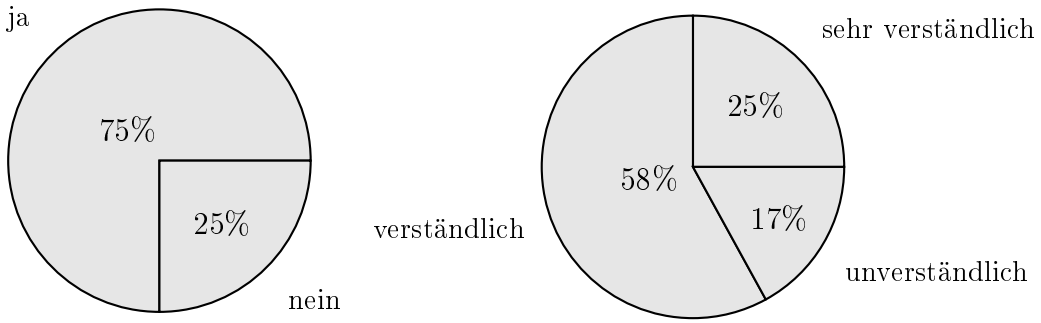
(a) Position der Steuerelemente

(b) Gestaltung der Oberfläche

Abbildung 35: Zufriedenheit mit der Oberfläche

Das vertraute Gerät führte dazu, dass der Roboter und seine Fähigkeiten gut eingeschätzt wurden und ihm sogar ein gewisses Vertrauen entgegen gebracht wurde. Zu einer Fehleinschätzung der Fähigkeiten kam es nicht.
Keiner der Testpersonen hatte den Eindruck, dass der Roboter Cosero Treppen steigen könnte, dafür waren fast alle davon überzeugt, dass er Gegenstände vom Boden aufheben kann (siehe Abbildung 37).

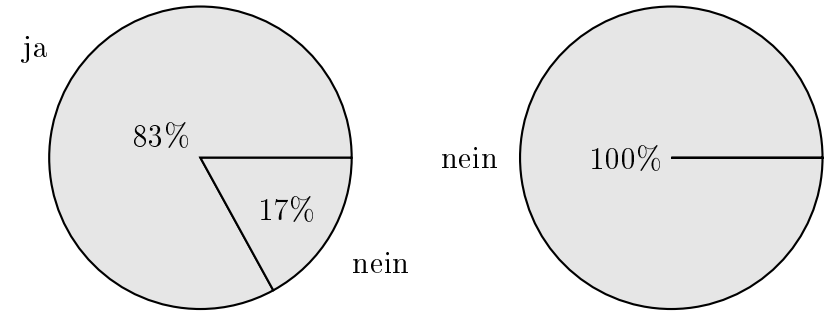
Die Testpersonen haben darauf hingewiesen, dass ihnen die Ausrichtung der Kamera nicht zu jedem Zeitpunkt klar war (siehe Abbildung 38a). Die Ausrichtung der Kamera ließe sich einfach als zusätzliche Information im Bildschirm visualisieren.
Überwiegend zufrieden waren die Testpersonen mit dem Angebot der verschiedenen Autonomiestufen und der unterschiedlichen Steuermöglichkeiten. Der Wechsel zwischen den verschiedenen Funktionen sei einfach durchzuführen (siehe Abbildung 38b).



(a) Benutzung des Eingabegeräts führt zu erwartetem Ergebnis

(b) die verwendeten Begriffe sind

Abbildung 36: Bedienbarkeit der Steuerelemente



(a) Gegenstände vom Boden greifen

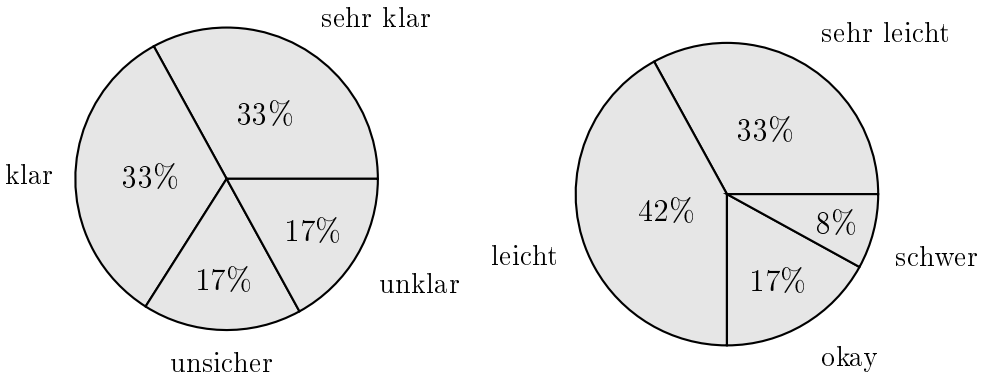
(b) Treppen steigen

Abbildung 37: Fähigkeiten des Roboters

6.3.5 Integration

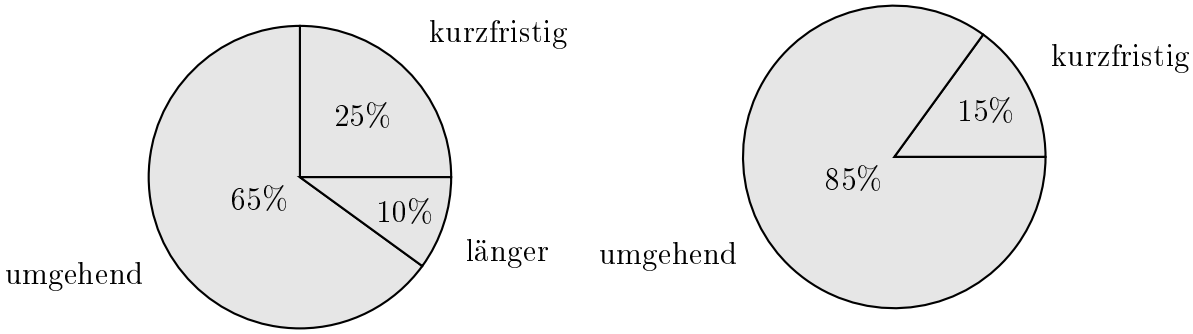
Die befragten Testpersonen, die ein Android-Gerät besitzen, gaben verstärkt an, dass sie ihr Handheld immer bei sich tragen und durchschnittlich innerhalb weniger Minuten auf eine Benachrichtigung reagieren. SMS werden von den meisten Benutzern innerhalb von 10 Minuten gelesen, während auf Anrufe meist umgehend reagiert wird (siehe Abbildung 39a). Interessant ist, dass die Testpersonen zwischen verschiedenen Klassen von Anrufen unterscheiden (siehe Abbildung 39). Es ist daher anzunehmen, dass wenn der Roboter einen Hilferuf sendet oder um Aufmerksamkeit bittet auch umgehend Hilfe erhält.

Im Gegensatz zu anderen zusätzlichen Geräten sind Android-Geräte entweder schon im Haushalt etabliert oder würden sich schnell integrieren. Die Nutzer schenken dem Handheld genügend Aufmerksamkeit, sodass der Roboter ausreichend interagieren



(a) Kameraausrichtung (b) Funktionenwechsel

Abbildung 38: Zufriedenheit mit der Anwendung



(a) Telefonanruf (b) wichtiger Telefonanruf

Abbildung 39: Reaktionszeit auf Handybenachrichtigungen

kann. Dies vereinfacht die Bedienbarkeit und die Integration eines Service-Roboters in einen Haushalt.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurde eine Android-Anwendung zur Teleoperation eines persönlichen Service-Roboters entwickelt. Mit dieser Anwendung ist es möglich den Roboter auf drei unterschiedlichen Autonomiestufen zu operieren. Für diese Aufgabe stehen Roboter und Anwendung dauerhaft über eine drahtlose Netzwerkverbindung in Kontakt. Eine Vorverarbeitung von Messdaten wird eingesetzt, um die notwendige Bandbreite zur Teleoperation gering zu halten und Schutzmaßnahmen anzubieten. Das Ziel ist den Anwender maximal zu entlasten und dabei eine künstliche Einschränkung der Fähigkeiten des Roboters zu vermeiden. Verfolgt wird ein Konzept der anpassbaren Autonomie, in welcher Anwender und Roboter die Beziehung eines gemischten Teams eingehen.

Für einfache, aber spezielle, bis hin zu komplizierten Aktivitäten wird über die erste Autonomiestufe eine abgesicherte Teleoperation ermöglicht. Der Anwender kontrolliert den Roboter über kontinuierliche Steueranweisungen. Die Absicherung unterstützt den Anwender und sorgt für eine Gefahrenvermeidung. Für Ausnahmesituation ist es möglich auch diese Einschränkung zu umgehen, um bei einer unzureichenden Wahrnehmung den Roboter zu unterstützen und aus einer für ihn ausweglosen Situation zu befreien. Bedient werden kann in dieser Stufe sowohl der omnidirektionale Antrieb, als auch die Blickrichtung und die Endeffektorpose des anthropomorphen Arms.
In der zweiten Autonomiestufe, dem halbautonomen Modus, werden sogenannte Aktionsprimitive zur Verfügung gestellt. Sie erlauben die autonome Ausführung einfacher Aktionen, die regelmäßig im Betrieb des Roboters notwendig sind. Der Fokus liegt auf einer minimalen Interaktionszeit. Die manuelle Ausführung dieser Aktionen würde für den Anwender eine überdurchschnittliche Belastung bedeuten. Implementiert wurde die Ausrichtung des Blicks und die Navigation zu einem im Kamerabild gewählten Punkt. Darüber hinaus können in einem weiteren Aktionsprimitiv über eine Objektsegmentierung erkannte Gegenstände auf einem Tisch ausgewählt und gegriffen werden. Das Aktiosprimitiv der Navigation auf einer Karte dient zur einfachen Erreichbarkeit beliebiger Orte.
In der dritten Autonomiestufe erlaubt die Anwendung die Definition einer vollständigen Aufgabe, die der Roboter gänzlich autonom ausführt. Treten Probleme auf oder fehlt dem Anwender Funktionalität, so besitzt er die Möglichkeit über niedrigere Autonomiestufen das gewüschte Ziel zu erreichen.

Bestehende Konzepte und Empfehlungen aus der Mensch-Roboter-Interaktion wurden beschrieben und eingesetzt, um dem Anwender eine leichte Interaktion durch eine natürliche Darstellung zu erlauben. Mit Hilfe einer Erweiterung der Realitätswahrnehmung, durch die perspektivische Projektion von Objekten in das Kamerabild, wurde eine leichte Erlern- und Bedienbarkeit erzielt.

An Hand einer Benutzerstudie unter Berücksichtigung eines Anwenderszenarios wurde die Anwendung evaluiert und das Konzept der Steuerung bewertet.

Es wurde beobachtet, dass die Benutzer zwar Spaß hatten die Fähigkeiten des Roboters spielerisch zu erlernen, jedoch nur einen Mehrwert erfahren, wenn primitive Aufgaben durch den Roboter autonom gelöst werden. Besonders intuitiv in diesem Zusammenhang wurde im Gegensatz zur Joysticksteuerung die Steuerung mit Gesten wahrgenommen.

Ausblick In der Mensch-Computer-Interaktion wird die Einführung von Gesten als Bedienelement kritisch betrachtet. Für herkömmliche Bedienelemente existieren gut getestete und verstandene Standards. Seit dem Einsatz von Gesten werden diese Standards ignoriert, verworfen oder verletzt. Eine Übertragung von etablierten Richtlinien auf Gesten ist ausgeblieben. Auch Marktführer, wie Apple oder Google ignorieren alte Richtlinien und unternehmen den Versuch schlecht umgesetzt neue Richtlinien einzuführen.
Ursprünglich wurden grafische Benutzeroberflächen geschaffen, um dem Anwender das Einprägen komplizierter Befehle zu ersparen. Die gesamte Funktionalität einer Anwendung konnte erschlossen werden, indem der Anwender die Menüstruktur der Anwendung untersuchte. Dieses Prinzip wird in der HCI als Auffindbarkeit bezeichnet. Auf Smartphones verschwindet dieses Prinzip zunehmend. Apple hat dabei begonnen vollständig auf Menüs zu verzichten. Google hat mit Android bisher keine klare Position bezogen. Zu Anfangs wurde ein permanenter Menü-Button angeboten, welcher sich jedoch als irreführend herausgestellt hat, wenn er nicht mit einer Funktion belegt ist. Für Gesten existiert bis heute kein Konzept für eine Gewährleistung der Auffindbarkeit. Es müssen Wege geschaffen werden, um den Awender über dieses Feature zu informieren und Gesten sowohl sichtbar als auch auffindbar zu machen. Eine häufige Reaktion beim Auffinden einer Geste ist die Überraschung. Oft bedarf es dazu eines weiteren Anwender, der die Möglichkeiten einer magischen Geste vorführt.

Eine fehlende Konsistenz beim Einsatz von Gesten erschwert die Verwendung zusätzlich. Früher waren Bedienkonzepte eindeutig definiert. Felder zum Ankreuzen lassen eine Mehrfachauswahl zu, Optionsfelder die Auswahl einer Möglichkeit. Auf einem Smartphone kann der Entwickler aktuell die Funktion und den Einsatzzweck eines Bedienelements frei wählen. Es können Gesten zur Skalierung von Bildern eingesetzt werden oder Knöpfe. Ein Druck auf den Bildschirm kann unzählige Aktionen ausführen. Häufig wird dabei ein Bild vergrössert, ein Hyperlink geöffnet oder ein Bild rotiert. Es werden zwar Human Interface Guidelines angeboten, jedoch verfolgen viele Unternehmen ihre eigenen Vorstellungen. Dies hat sogar zur Folge, dass spezielle Bedienkonzepte geschützt sind und von anderen Unternehmen nicht eingesetzt werden können. Ein Beispiel für ein solches US-Patent ist die Nummer 5946647. Es handelt sich dabei um ein Verfahren, welches unter anderem Telefonnummern in E-Mails und Internetseiten anklickbar macht, um einen Anruf zu erleichtern. Proprietäre Standards zerstören auf diese Weise eine systemübergreifende konsistente Bedienung.

Auch fehlen bis heute Richtlinien für die bildschirmunabhängige Imple-

mentierung von Gesten. Es kann keine Aussage darüber getroffen werden, wie sich Gesten im Verhältnis zur Bildschirmgröße verhalten sollen. Gesten für kleine Bildschirme können auf Großen Probleme bereiten und umgekehrt. Es ist möglich, dass es angenehmer ist eine Geste in einer Bildschirmecke auszuführen, da man das Geräte mit beiden Händen hält. Andere Anwender halten ein Tablet in einer Hand, während die andere Hand für weitreichende Interaktionen zur Verfügung steht. Eine weitere Gruppe legt das Gerät auf den Tisch oder seine Beine, um es zu bedienen. Es besteht bis heute kein Konzept, wie eine sinnvolle Bedienung erfolgen sollte. Gleiches gilt für die Lage an sich. Diese Problematik ist auf Smartphones übertragbar.

Die Zuverlässigkeit ist ein weiteres Problem beim Einsatz von Gesten. Sensitive Bildschirme bürgen die Gefahr, dass versehentlich Aktionen ausgeführt werden. Auf kleinen Bildschirmen, bei welchen die Bedienelemente nahe beisammen liegen, steigt das Risiko einer fehlerhaften Bedienung. Aber auch einem Tablet wird leicht versehentlich eine Aktion ausgeführt beim Versuch das Gerät zu halten oder zu stabilisieren. Versehentliche Aktionen sind eine gängige Praxis bei dieser Art von Geräten. Auch das nicht ausreichende Berühren und das Ausbleiben einer Aktion kann die Zufriedenheit beeinträchtigen. Durch die Unsichtbarkeit von Gesten ist die Ursache einer ungewollten Aktion nur schwer nachvollziehbar. Auch das Reproduzieren eines Fehlers erweist sich als kompliziert. Eine Großzahl der Probleme, die bei der Benutzung auftreten sind keine Bedienfehler des Anwenders, sondern ein Fehler des Entwicklers, der es zu einfach gemacht hat einen Fehler zu begehen. An einem Desktop und dem Einsatz einer Maus können ähnliche Probleme auftreten, jedoch erlauben Mauszeiger und ein eindeutiges Feedback der Maustaste eine leichtere Identifizierung von Problemen. Das Auslösen von zufälligen Aktionen wirkt in jedem Fall verstörend und ermüdend.

Desweiteren wird sich der Markt für die beschriebene Robotik-Anwendung in den nächsten Jahren weiter verschieben. Weg von Personengruppen, die auf Hilfe angewiesen sind und für die Ausführungsgeschwindigkeit einer Aufgabe unerheblich ist, hin zu Personen die unangenehme oder lästige Aufgaben an einen Roboter abgeben möchten. Die offensichtlichen Schwächen des Beispielsszenarios in Zusammenhang mit Sehvermögen und Koordinationsfähigkeit wird sich ab diesem Zeitpunkt erübrigen. Eingabegeräte werden entsprechend bekannt und erprobt sein. Denkbar ist auch, dass ein Fernseher statt einem Tablet als Eingabegerät zum Einsatz kommt. Ein Google-TV auf Android-Basis ist seit längerem in den Schlagzeilen.

Fazit Diese Arbeit hat sich mit den Grundlagen der Mensch-Roboter-Interaktion unter Verwendung eines Service Roboters und eines Android-Handhelds beschäftigt. Es wurde Software entwickelt, die das einfache bedienen des Roboters in verschiedenen Autonomiestufen ermöglicht. Die Testergebnisse und die Entwicklungen auf dem Technikmarkt liefern eine Vielzahl von Möglichkeiten die Anwendung weiter zu entwickeln und vielseitig einzusetzen.

Literatur

[Alb91] Albus, J. S.: *Outline for a Theory of Intelligence.* IEEE Transactions on Systems, Man, and Cybernetics, 21:473–509, 1991.

[Beh] Behnke, S.: *Autonomous Intelligent Systems, NimbRo @Home.* `http://www.ais.uni-bonn.de/nimbro/@Home/`.

[Bil00] Billard, A. und Dautenhahn, K.: *Experiments in social robotics: grounding and use of communication in autonomous agents.* Adaptive Behavior, 7(3/4), 2000.

[BRJ+11] Bohren, J., R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen und S. Holzer: *Towards autonomous robotic butlers: Lessons learned with the PR2.* In: *ICRA*, Seiten 5568–5575, 2011.

[CA07] Clarkson, E. und R. C. Arkin: *Applying Heuristic Evaluation to Human-Robot Interaction Systems.* In: *FLAIRS Conference'07*, Seiten 44–49, 2007.

[CBS] CBS: *Google unveils cell phone software and alliance.* `http://news.cnet.com/8301-17939_109-9810937-2.html`.

[CG01] Crandall, J. W. und M. A. Goodrich: *Experiments in adjustable autonomy.* IEEE International Conference on Systems, Man, and Cybernetics, 3:1624–1629 vol.3, 2001.

[CMN83] Card, S. K., T. P. Moran und A. Newell: *The psychology of human-computer interaction.* CRC, 1983.

[DKB+08] Dias, M. B., B. Kannan, B. Browning, E. Jones, B. Argall, M. F. Dias, M. B. Zinck, M. M. Veloso und A. T. Stentz: *Sliding Autonomy for Peer-To-Peer Human-Robot Teams.* System, 2008.

[DN90] Dorf, R. C. und S. Y. Nof (Herausgeber): *Concise International Encyclopedia of Robotics: Applications and Automation.* John Wiley & Sons, Inc., New York, NY, USA, 1st Auflage, 1990.

[Dru04] Drury, J.L.: *Design Guidelines for Improved Human-Robot Interaction.* In: *in CHI Extended Abstracts 2004*, Seite 1540. ACM, 2004.

[DSK07] Drury, J. L., J. Scholtz und D. E. Kieras: *Adapting GOMS to model human-robot interaction.* Seiten 41–48, 2007.

[Foua] Foundation, Eclipse: *The Eclipse Foundation open source community website.* `http://www.eclipse.org/`.

[Foub] Foundation, Free Software: *GNU Emacs.* `http://www.gnu.org/software/emacs/`.

[Gara] Garage, Willow: *Roboter PR2.* http://www.willowgarage.com/sites/default/files/robots_pr2/Beta_1-5.jpg.

[Garb] Garage, Willow: *ROS community website.* http://www.ros.org/wiki/.

[Ger08] Gertman, D. I. und Bruemmer, D. J.: *Human factors principles in design of computer-mediated visualization for robot missions.* In: *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*, Seiten 237 –242, 2008.

[GG08] Gunderson, J. P. und L. F. Gunderson: *Intelligence ≠ Autonomy ≠ Capability.* 2008.

[GO03] Goodrich, M. und D. Olsen: *Seven principles of efficient human robot interaction.* In: *International Conference on Systems*, 2003.

[GS07] Goodrich, Michael A und Alan C Schultz: *Human-Robot Interaction: A Survey.* Foundations and Trends in Human Computer Interaction, 1(3):203–275, 2007.

[HHE04] H. Hüttenrauch, A. Green, M. Norman L. Oestreicher und K.S. Eklundh: *Involving users in the design of a mobile office robot.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 34(2):113 –124, Mai 2004.

[Hüt06] Hüttenrauch, H.: *From HCI to HRI: Designing Interaction for a Service Robot.* Doktorarbeit, KTH, Sweden, 2006.

[Jäh05] Jähne, B.: *Digitale Bildverarbeitung.* Springer, 2005.

[Koha] Kohler, D.: *Rosjava - An implementation of ROS in pure-Java with Android support.* http://www.ros.org/wiki/rosjava.

[Kohb] Kohler, D.: *Scripting Layer for Android.* http://code.google.com/p/android-scripting/.

[Mil07] Miller, C. A.: *Designing for Flexible Interaction Between Humans and Automation: Delegation Interfaces for Supervisory Control*, 2007.

[Mur00] Murphy, R. R.: *Introduction to AI Robotics.* MIT Press, Cambridge, MA, USA, 1. Auflage, 2000.

[NGR07] Nielsen, C. W., M. A. Goodrich und R. W. Ricks: *Ecological Interfaces for Improving Mobile Robot Teleoperation.* IEEE Transactions on Robotics, 23(5):927–941, 2007.

[OJC+11] Osentoski, S., G. Jay, C. Crick, B. Pitzer, C. DuHadway und O. C. Jenkins: *Robots as web services: Reproducible experimentation and application development using rosjs.* In: *Proccedings of the 2011 IEEE International Conference on Robotics and Automation*, 2011.

[oR] Robotics, IFR International Federation of: *Service Robot Statistics*. http://www.ifr.org/service-robots/statistics/.

[Par] Parrot: *Ar.Drone*. http://ardrone.parrot.com.

[PLKB08] Park, K.-H., H.-E. Lee, Y. Kim und Z.Z. Bien: *A Steward Robot for Human-Friendly Human-Machine Interaction in a Smart House Environment*. IEEE T. Automation Science and Engineering, 5(1):21–25, 2008.

[PR] PR, Markengold: *Parrot AR.Drone*. http://www.markengold.de/news/parrot-ar-drone-flugerlebnisse-erhohen/.

[Proa] Project, Android Open Source: *Backstack, Android Developers Guide*. http://developer.android.com/images/fundamentals/diagram_backstack.png.

[Prob] Project, Android Open Source: *Developers Guide*. http://developer.android.com/images/activity_lifecycle.png.

[Proc] Project, Android Open Source: *Fragments, Android Developers Guide*. http://developer.android.com/guide/topics/fundamentals/fragments.html.

[Prod] Projects, Poignant: *Androvio Blog*. http://4.bp.blogspot.com/-piS_HB3ra2I/TjbsnmGOwiI/AAAAAAAAAY0/E8Oo07Ldz7Q/s1600/n1-portrait_androvio.png.

[PSW00] Parasuraman, R., T. B. Sheridan und C. D. Wickens: *A model for types and levels of human interaction with automation*. IEEE Transactions on Systems Man and Cybernetics Part A Systems and Humans, 30(3):286–297, 2000.

[QBN07] Quigley, M., E. Berger und A. Y. Ng: *STAIR: Hardware and Software Architecture*, 2007.

[QGC+09] Quigley, M., B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler und A. Ng: *ROS: an open-source Robot Operating System*. In: *International Conference on Robotics and Automation*, Nummer 1, 2009.

[RDIZ03] Riva, G., F. Davide, W. A Ijsselsteijn und Shangyang Zhao: *„Being there“ and the Role of Presence Technology*, 2003.

[SCG09] Sung, J., H.I. Christensen und R.E. Grinter: *Robots in the wild: understanding long-term use*. In: *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, HRI '09, Seiten 45–52, New York, NY, USA, 2009. ACM.

[Sch07] Schreer, O.: *Stereoanalyse und Bildsynthese.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[SDKP06] Seffah, A., M. Donyaee, R. B. Kline und H. K. Padda: *Usability measurement and metrics: A consolidated model.* Software Quality Control, 14:159–178, Juni 2006.

[She92a] Sheridan, T. B.: *Musings on telepresence and virtual presence.* Presence: Teleoper. Virtual Environ., 1:120–126, Januar 1992.

[She92b] Sheridan, T. B.: *Telerobotics, Automation, and Human Supervisory Control.* The MIT Press, August 1992.

[Ste92] Steuer, J.: *Defining Virtual Reality: Dimensions Determining Telepresence.* Journal of Communication, 42:73–93, 1992.

[Thr04] Thrun, S.: *Toward a framework for human-robot interaction.* Human-Computer Interaction, 19:2004, 2004.

[Wik] Wiki, ROS: *ROS Wiki.* `http://mirror.umd.edu/roswiki/ApplicationsPlatform%282f%29ApplicationsPlatformOverview.html`.

[Wow] WowWee: *Rovio.* `http://www.wowwee.com/en/products/tech/telepresence/rovio/rovio`.

Abbildungsverzeichnis

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Diplomarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine außer den angebenen Quellen und Hilfsmittel verwendet, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Sebastian Muszynski